Quaternion-based Human Gesture Recognition System Using Multiple Body-worn Intertial Sensors

by

Shamir Alavi

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

Master of Applied Science
in
Electrical and Computer Engineering (Data Science)
Ottawa-Carleton Institute for Electrical and Computer Engineering

Carleton University
Ottawa, Ontario, Canada

Abstract

In this study, we designed a multi-sensor gesture recognition system that can classify among six different human gestures. Data was collected from eleven participants using five gyroscopic motion sensors tied to their upper body. A total of 1080 samples were collected, which contain almost 6000 gestures collected within a span of 90 minutes. The data were processed and fed into a multiclass Pattern Classification system to classify the gestures. We trained Support Vector Machines and Artificial Neural Networks on the same dataset under two different scenarios to compare the results. A similar study was performed before using modified Hidden Markov Model but the data was collected using a single sensor. Our study indicates that near perfect classification accuracies are achievable. However, such accuracies are more difficult to obtain when a participant does not participate in training even if the test set does not contain any data from the training set.

Acknowledgements

I would like to extend my heartfelt gratitude to my supervisor, Dr. Anthony Whitehead, who not only was very calm and collected, patient, amicable, and informative during the weekly discussions but also gave me enough space to steer the research in my own direction.

I would like to acknowledge the infinite patience and tolerance of my wife, Elora. It was for her endless support that I was able to work even during weekends and late nights in the lab trying to finish my research.

I am also thankful to my former colleague, Dennis Arsenault, whose work on finding out the issues with the sensors made my job easier in concentrating on the actual research.

Table of Contents

List of Tables

List of Figures

List of Appendices

List of Acronyms

SVM - Support Vector Machines

ANN - Artificial Neural Networks

PCA - Principal Component Analysis

P – Participants

Tr – Training

CV: Cross-validation

1 Chapter: Introduction

In today's fast-paced world, accuracy and efficiency is of paramount importance to any industry, be it for manufacturing and technology or for entertainment. Human activity is either getting aided by assistive technologies for better and faster performance, or being replaced by automated systems to obtain higher accuracy. While we still have a long way to go to build computers as intelligent as humans, we have come a long way in recognizing how the human body works and create machines that can imitate the gestures to perform repetitive works with more speed and zero fatigue.

Human gesture recognition has long been a considered a solution to such problems. The physical nature of our body limits us from being consistently efficient in performing long, repetitive tasks. Be it in the real world or in virtual reality, increasing our work efficiency while being able to perform tasks accurately is the problem that we have been trying to solve for years. One solution is to build machines which can recognize our gestures and carry out what we intend to perform. With fast modern day microprocessors and computers, it is the correct time to work on building comprehensive gesture recognition systems.

1.1 Gesture Recognition

Gestures are physical movements of different parts of the body that are expressive and meaningful to every living being. We perform gestures to convey meaningful information or to interact with the environment. Gesture recognition has a wide variety of applications that includes but is not restricted to developing aids for the hearing impaired, recognizing sign language, navigating in virtual environments, automation of manufacturing industry etc. [1]. There are various techniques that can be used to recognize gestures, ranging from using mathematical models based on Hidden Markov Model [2], Hidden Markov Chain or

other soft computing tools [1] to applying Computer Vision based approaches [1][3] by using data gloves [4] and accelerometers [5][6], or using a combination of both [1]. Gestures can be hand and arm gestures, head and face gestures or full body movements [1].

1.2 Thesis Overview

The thesis starts with a discussion in Chapter 2 on the background of the topics related to our study. Here, we discuss the operating principles, types and applications of body area sensors. These are an integral part to almost all of the gesture recognition research. Then we discuss briefly about Data Analysis, its relevance in today's world and its application in our research. After that, we talk about the Pattern Classification or Machine Learning algorithms that we used to analyze our dataset. We discuss about the operating principles of Support Vector Machine and Artificial Neural Network, and talk about the rationale behind selecting these learning algorithms. Finally, we explain the Feature Selection and Dimensionality Reduction methods that were used to carry out our analysis.

In Chapter 3, we present a literature review of previous studies that were done on this topic, show where the state of the art is right now and describe how our study is relevant to the current literature. Chapter 4, which is the main chapter of this thesis, focuses on our research/experimental methodology in detail. We start with describing the physical nature of the experiment in the first subsection of this chapter. Here, we provide a short summary of the sensors that we used for data collection, an explanation of the quaternion coordinate system that the sensors use and how we converted raw quaternion values into Euler angles. Then we briefly discuss about sensor drift and sensor axis alignment. We also mention how we placed the sensors on the body, the data transfer protocol from sensor to computer, the

set of gestures that the participants were asked to perform and finally the limitations of our study.

In the next subsection of Chapter 3, we get into the details of our data analysis process. We start by stating which Machine Learning tools, libraries and programming languages were used to carry out this research. Then we explain how we partitioned and preprocessed the data, where we also talk about a modified noise filtering method that we developed to remove unnecessary noise or outliers from the dataset. The next subsection explains our feature extraction process in which we provide details on two kinds of feature sets: single sensor-based features and multiple-sensor based features.

Our discussion then moves on to the training phase of our study. We start by performing an initial experiment that reveals the weaknesses in our initial approach. This, in turn, leads us to applying dimensionality reduction and feature selection techniques after carefully looking at the performance of the classifiers on the cross-validation set. The final section of this chapter shows the results of applying our classification model on two independent test sets and presents the corresponding classification accuracies.

The final chapter, Chapter 5, summarizes our experimental methodology and test results, the applications and limitation of our study and discusses relevant future works that can be done to improve upon our findings.

1.3 Research Questions

The previous research on this topic validated the accuracy of this type of gesture classification system using a single sensor as a baseline.It used Hidden Markov Model and Modified Markov Chain [1][7] for classification. Following up with that, we are posing two main research questions. The first one asks if multiple sensors improve recognition

accuracy and the second one compares the accuracy between the two classification algorithms to find out which one would be the most suitable to carry forward this research.

1.4 Study Overview

To address the research questions, we conducted a study with 11 participants. Each of them wore five sensors on their body – 2 on both the arms, 2 on both the wrists and 1 on the upper abdomen. They were asked to perform 6 gestures, namely Jab, Uppercut, Throw, Lift, Block and Sway. As they performed the gestures, the sensors transmitted the data in quaternion forms to a computer through a wireless protocol. The data was stored in CSV format for further analysis. Since our system is not built on a kinematic model, the constraints of the kinematic system, such as exact position of the sensors on the participants' body was not significant to our study. However, the relative positioning of the devices, e.g. one sensor on the arm and another on the wrist, was important for calculating features.

1.5 Applications in the Real World

Since the advent of intelligent and interactive computing, gesture recognition has been a prevalent approach to establish interactions between humans and computers [1]. It is a common application of motion capture and it is mainly performed using wearable or body-worn sensors [8][9][10]. Body-worn sensors are used in interactive gaming controller, such as Nintendo™ Wii, to capture users' motions and recognize their gestures in-game [11]. Similar applications on augmented and virtual reality use hand-gesture recognition or capture motion data from the body based on either wearable-sensor based [12] or vision-based systems [13]. On the other hand, different industries, such as the medical (health) industry and the manufacturing industry have been using this technology to employ robot-

assisted living [14] and improve industrial robotics [15].

1.6 Contributions

The contribution of our research is threefold:

- Creation and publication of a complete dataset of 11 participants containing IMU data for 6 different gestures.
- Development of a comprehensive classification system to recognize these gestures.
- Analysis of multi-sensor classification and its comparison with the performance of single sensor-based classification system.

2 Chapter: Background

2.1 Body Area Sensors

Body area sensors determine the current physical state, or changes in such states of an individual. Typically, these are used in a sensor network called Body Area Sensor Network (BASN) [16]. These sensors are typically worn on the body. In a few special cases, they may also be implanted within the body. Figure 1 shows a basic design of such a network and how information is transferred from an individual to the data processing unit.

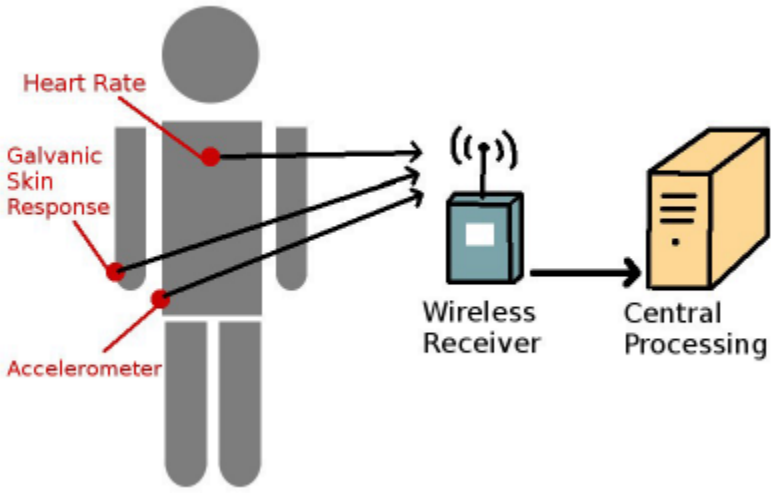


Figure 1: A theoretical Body Area Sensor Network (BASN) [7]

With recent technological advances, sensors have become much smaller with the development of Micro-Electro-Mechanical Systems (MEMS) [17], they have faster processing times, can operate on batteries, can communicate wirelessly and have easier wearability. Sensors have applications in a wide variety of fields. In terms of applications, we can roughly divide them into two principle categories – Physiological and Biomechanical.

Physiological sensors detect the state changes in physiological properties of the body. Heart Rate Sensor [18], Galvanic Skin Response Sensor (GSR) [19], Electromyography Sensor

[20], Brain Computer Interfaces (BCI) [21], Respiration Sensor [22] are but to name a few. Mechanical sensors are used to measure the physical position and movements of an individual. This is the type of sensor that has been used throughout this thesis. Out of all the different types of biomechanical sensors, accelerometers, gyroscopes and magnetometers are the ones that are significant to this study.

Accelerometers: An accelerometer can measure acceleration that occurs along a device's axes. 3-axis accelerometers are used now-a-days so that acceleration can be measured in any direction. Most modern accelerometer designs place the three accelerometers in an orthogonal configuration to achieve this. They provide information on their angle of inclination with respect to downward direction by sensing the acceleration of gravity. However, these sensors are unable to distinguish between gravitational force and actual accelerations [23]. They suffer most from noise in their readings.

Gyroscopes: Gyroscopes are used to detect angular velocities. However, they are unable to determine their absolute orientation and suffer from drift issues [24]. Drift adds small angular velocities even when the device is completely stationary. Over time, it accumulates and affects the overall reading of the sensor. This issue has been specifically addressed in section 3.1.5.

Magnetometers: Magnetometers are used to discern the direction of earth's local magnetic field. They use earth's field as a reference direction. As earth's magnetic field is quite weak, one thing to be careful about while using magnetometers is to keep them away from nearby metallic objects to prevent magnetic interference [24].

Using accelerometers, gyroscopes and magnetometers together can produce better measurements than that from any of these sensors individually. This combination is

produced using the Sensor Fusion algorithm [25]. The device is often collectively called an Inertial Measurement Unit (IMU) [26]. Section 3.1.1 discusses this in detail.

2.2 Coordinate System: Quaternions

The sensors that we used in this study output their orientations in the form of quaternions. A quaternion is comprised of a scalar component and a vector component in complex space. This representation can be seen in the following equations:

$$q = (r, \vec{v}) \quad \text{..... (1)}$$

$$= (q_r, q_x\hat{\imath}, q_y\hat{\jmath}, q_z\hat{k}) \quad \text{..... (2)}$$

Here $\hat{\imath}, \hat{\jmath}$ and $\hat{k}$ are complex orthogonal basis vectors which makes quaternion multiplication a non-commutative operation. It means $q_x\hat{\imath} * q_y\hat{\jmath} \neq q_y\hat{\jmath} * q_x\hat{\imath}$ (similar rule for addition). Full multiplication result for two general quaternions 'a' and 'b' are shown in equation 9. Our goal in this section is provide adequate knowledge about quaternions so that the readers are able to understand the later sections. More details can be found in [27].

$$\begin{aligned} q = &\left(a_r, a_x i, a_y j, a_z k\right) * \left(b_r, b_x i, b_y j, b_z k\right) \\ &= (a_r b_r - a_x b_x - a_y b_y - a_z b_z, \\ &\left(a_r b_x + a_x b_r + a_y b_z + a_z b_y\right)\hat{\imath}, \\ &\left(a_r b_y + a_y b_r - a_x b_z + a_z b_z\right)\hat{\jmath}, \\ &(a_r b_z + a_z b_r + a_x b_y - a_y b_x)\hat{k}) \end{aligned} \quad \text{..... (3)}$$

All quaternions that represent a 3D rotation are represented by unit quaternions. A unit quaternion has a magnitude of 1 which means the absolute value or norm [28] of 'q' would be

$$|q| = \sqrt{{q_r}^2 + {q_x}^2 + {q_y}^2 + {q_z}^2} = 1 \quad \text{..... (4)}$$

Quaternions are extremely efficient at representing rotational and orientation information. A rotation of angle θ about a unit axis $\vec{n}$ can be represented as:

$$q_1 = (cos\frac{\theta}{2}, \vec{n}\, sin\frac{\theta}{2}) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

To rotate the current rotational state q_0 by the amount specified by q_1, we multiply it by q_1:

$$q_2 = q_1 q_0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (6)$$

This quaternion q_2 is equivalent to rotating a rigid body by q_0 and then rotating by q_1. A series of rotations can therefore be represented by a series of quaternion multiplications. It is a very efficient method of computing and representing a series of rotations. However, the order of these operations are very important due to the non-commutative property of quaternions.

Quaternions can be a powerful form of representing rotations over other forms of representations [27]. One of the biggest benefits of using quaternions is that they do not suffer from issues such as gimbal lock that methods like Euler Angles do. Moreover, they are very computationally efficient because they do not require the calculations of many trigonometric functions. On the contrary, they suffer from being more conceptually difficult to understand and more abstract to visualize.

2.3 Pattern Classification and Data Analysis

Pattern classification involves the use of (pattern) classifiers to distinguish among different, interesting patterns inherent in a dataset. In the broadest sense, it employs learning methods to find out meaningful patterns in the data [29]. Usually, the whole dataset is divided into two or three smaller subsets to use them in training the classifier, cross validating the training set results (if we have three subsets) and finally testing the model on the final

subset of data. Learning refers to the use of some form of algorithm to reduce error on the training portion of the dataset. It comes in several forms such as supervised learning, semi-supervised learning, unsupervised learning, representation learning etc. Commonly used algorithms fall under the first two categories.

Supervised Learning: In this form of learning, category labels and costs of making errors in recognizing each pattern is provided in a training set. The goal is to eventually reduce the sum of the cost for these patterns [29]. During the process, important features are extracted from the data, redundant features are excluded if necessary and a few other tunings such as noise removal, feature scaling etc. are performed to minimize the training cost in order to achieve the best possible outcome in the testing stage.

Unsupervised Learning: In this case, no category labels or training labels are provided to the classifier. The system mainly works by forming 'clusters' or natural groupings of the input patterns. Given a particular set of patterns or cost functions, different clustering algorithms lead to different clusters. Often, the user can set the required amount of clusters ahead of time [29].

Several data analysis methods such as Correlation Analysis, Regression Analysis, Data Mining etc. are used to analyze data. While each method has its own area of application, we are interested in Data Mining for this study. It is used to accomplish six different tasks: Classification, Estimation, Prediction, Association Rules, Clustering and Visualization [30]. Gesture recognition, which is basically recognizing or classifying (human) gesture patterns, is a (pattern) Classification task.

We will briefly go over the theories of the two supervised learning algorithms that are relevant to our discussion and outline their strengths and weaknesses as classifiers.

2.3.1 Support Vector Machines

Support Vector Machines (SVM) is a machine learning algorithm that identifies separator(s) [31] i.e. it separates data into two (binary classification) or more (multiclass classification) groups or classes. In a broader sense, the goal of this classifier is to assign two (or more) vectors in the data to two (or more) distinguishable classes. These are called the support vectors. Then the classifier assigns other vectors in the dataset to one of these classes using a linear separator(s). The core idea is to define an optimization problem based on the separated support vectors. The optimization goal is to minimize the classification error and maximize the geometric margin between two linearly separable classes. This is why SVM is also known as 'maximum margin classifier' or 'wide margin classifier' [29]. The complete mathematical details of the SVM algorithm, from training to classification, is beyond the scope of this thesis. However, the basic outline of the algorithm as a binary classifier is given below:

- Given training data (x_i, y_i) for $i = 1 \dots N$, with $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, learn a classifier $f(x)$ such that

  $$f(x_i) = \begin{cases} \geq 0, & y_i = +1 \\ < 0, & y_i = -1 \end{cases} \qquad \dots\dots (7)$$

  i.e. $y_i f(x_i) > 0$ for a correct classification.

- Linear classifier: A linear classifier has the form:

  $$f(x) = w^T x + b \qquad \dots\dots (8)$$

  In 2D, the discriminant is a line whereas, in 3D, the discriminant is a plane. w is normal to the line and b is the bias. w is known as the weight vector; it is learned using the training data and then discarded [32].

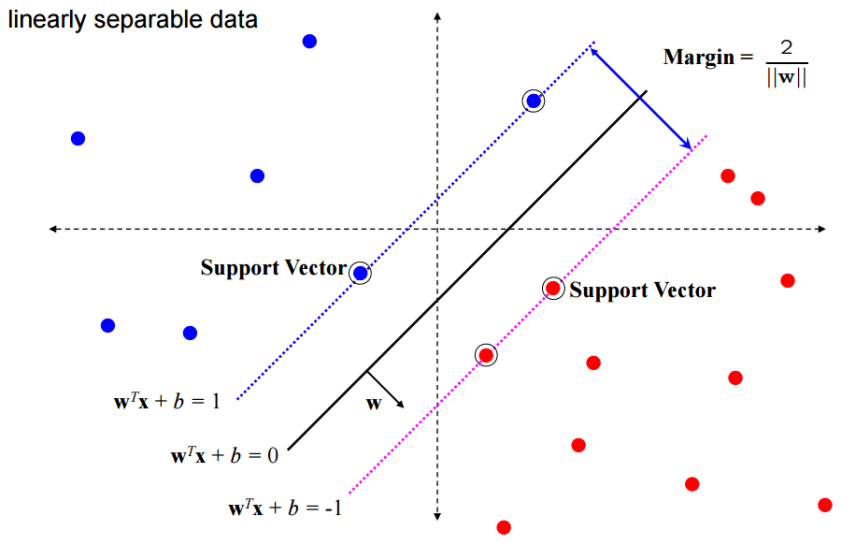


Figure 2: SVM as wide margin classifier
(Source: pemrogramanmatlab.wordpress.com/data-mining/support-vector-machine-2)

- $w^T x + b = 0$ and $c(w^T x + b) = 0$ on the discriminant boundary; thus the classifier has the freedom to choose the normalization for w. It chooses the normalization such that $w^T x_+ + b = +1$ and $w^T x_- + b = -1$ for the positive and negative support vectors respectively.
- The margin is given by,

$$\frac{w}{\|w\|} . (x_+ - x_-) = \frac{w^T(x_+ - x_-)}{\|w\|} = \frac{2}{\|w\|} \quad \text{..... (9)}$$

This margin can be optimized ($\max_w \frac{2}{\|w\|}$) to obtain the widest possible class separation.

To solve classification problems efficiently in very high dimensional feature spaces that are non-linear in nature, kernels are used to optimize the wide margin (often called as 'optimal margin classifier') [33][34]. Another important training feature is the regularization parameter that tells the SVM optimizer how much the user wants to penalize the classifier for each misclassification.

Multiclass classification: For a multiclass classification problem, a combination of SVM classifiers are trained as N one-versus-rest classifiers (i.e. 'one' positive, 'rest' negative) for the N-class case and that class is taken as the test point that corresponds to the largest positive distance from the rest [34].

One of SVM's strengths is that a user can avoid overfitting by properly choosing the regularization parameter. Overfitting occurs when a classifier usually fails to make generalizations regarding the underlying patterns in a dataset and has too many parameters relative to the number of observations in the dataset [35]. SVM is a great linear separator, but it also does well in multiclass problems as mentioned above. A disadvantage of SVM is model selection (choice of parameters) which may still end up causing overfitting.

2.3.2 Artificial Neural Networks

Artificial Neural Networks (ANN) are generalizations of mathematical models of biological nervous systems. The basic processing elements of an ANN are called artificial neurons or nodes. The functionalities of an actual neuron are thus represented by a simplified mathematical model where the effects of the synapses are represented by connection of weights that modulate the effect of the associated input signals. A transfer function represents the nonlinear characteristic shown by the neurons. The neuron impulse is then computed as the weighted sum of the input signal that is transformed by the transfer function. The network learns to adjust the weights in accordance to the chosen learning algorithm [36].

Architecture: In a typical neural network, as shown in Figure 3, the signal flows from inputs $x_1, x_2 \ldots x_n$ to the output is considered to be unidirectional. The neuron output signal is given by the following:

$$O = f(net) = f(\sum_{j=1}^{n} w_j x_j) \quad \text{..... (10)}$$

Where, w_j is the weight vector and the function is $f(net)$ is referred to as an activation (transfer) function. The variable net is defined as a scalar product of the weight and input vectors,

$$net = w^T x = w_1 x_1 + \cdots + w_n x_n \quad \text{..... (11)}$$

Here w^T is the transpose of matrix w, and, in the simplest case, the output value O is computed as,

$$O = f(net) = \begin{cases} 1, & if\ w^T x \geq \theta \\ 0, & otherwise \end{cases} \quad \text{.....(12)}$$

Where, θ is called the threshold level. θ can be selected using different linear or non-linear functions such as the step functions, the sigmoid function etc.

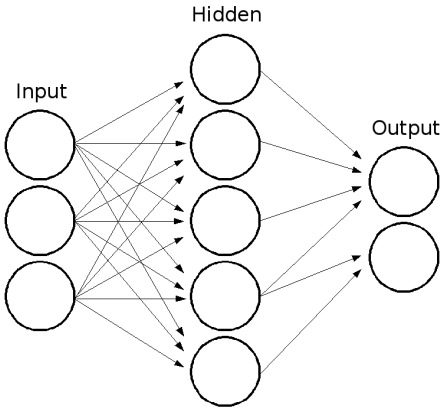


Figure 3: Basic architecture ofa multilayer neural network (Source: http://psychology.wikia.com/wiki/Artificial_neural_network)

The basic architecture of an ANN consists of three different types of neuron layers: input, hidden and output layers (Figure 3). A neural network is configured in such a way that the application of a set of inputs produces the desired set of outputs. There are different ways of doing this. One way is to set the weights explicitly, using a priori knowledge while another way is to train the neural network by providing certain patterns as inputs and letting

it change weights according to some learning rule [36]. As mentioned in the beginning of section 2.3, different learning situations can be used to train a neural network depending on the application and the type of data available. Below, we will review neural network training using backpropagation.

Backpropagation Learning: To understand the backpropagation algorithm, we need to know about feed-forward networks. In a feed-forward network, signal flows from the input to the output units, strictly in a feed-forward direction. The network may extend over multiple layers, but no feedback connections are provided. The backpropagation algorithm finds the local minimum of the error function by using the output of the feed-forward network as its input [37]. The network is initialized with randomly chosen weights. The gradient (or the rate of change) of the error function is computed and used to update the initial weights. The algorithm's task is to compute this gradient recursively.

Neural Networks can approximate any continuous function to any desired degree of accuracy which makes it useful for most classification problems. One of the drawbacks of ANN is that it takes much longer to train as compared to an SVM. On the other hand, using too many hidden layers may incite the problem of overfitting [38].

2.4 Feature Extraction

Most of the times, raw data that is collected during the initial phase of any Pattern Classification or Machine Learning experiment is not very meaningful to the classifiers. To perform well, classifiers need data that show good class separation [39]. This can be achieved by extracting meaningful features from the raw dataset. For most of the Pattern Classification systems, this is a very crucial step towards building an accurate and reliable classification model.

2.5 Feature Selection and Dimensionality Reduction

Sometimes the feature extraction process alone is not enough to feed the classifiers with meaningful data. This can happen due to several reasons. For example, a very common scenario is the extraction of highly correlated features such as length and volume or diameter and circumference. That is why even after feature extraction, if the classifiers show poor performance, we need to extract or create subsets of features that show the most variations or good class separation. This is called Feature Selection [39] which, in turn, also reduces the dimension of the feature dataset. Reducing the dimension of a feature set also enables faster computation. It can be done manually by selecting features that seem to show good class separation by visualizing the feature set. Sometimes different features originate from different sources (such as from different sensors in case of our study). Under this circumstance, we only select features from the source(s) that is/are most relevant to our dataset. However, it might not still be good enough for the classifiers. In that case, we need to apply better dimensionality reduction techniques that can take care of this issue. Principal Component Analysis (PCA) is one of them.

Principal Component Analysis (PCA): PCA, among many others, is a dimensionality reduction technique that is most commonly used in Pattern Classification. A thorough discussion on PCA is beyond the scope of this thesis and can be found in [40].

PCA projects multi-dimensional data into lower dimensional space or hyperspace but preserves necessary variations in the dataset. Sometimes it is confused with other feature selection methods but it is not necessarily a feature selection technique. As the name suggests, it derives Principle Components or PCs from a given set of features. Each PC preserves a certain variation of the data. Later, we can neglect the PCs that show relatively

less variation and construct a new feature set having lesser number of features but the useful ones. PCA works in the following way:

1. It calculates a covariance matrix that contains the covariance between every two features in the given dataset. This is how PCA knows how changes in one feature is associated with the changes in another.
2. Then it computes the Eigenvectors and corresponding Eigenvalues from the covariance matrix. Eigenvectors can be thought of as the axes (directions) along which a linear transformation acts. The transformations can be seen as simple stretch or compression. These vectors do not change directions when a linear transformation is applied to them. In this case, they tell us how appropriately each feature is projected onto a certain projection axis. The lesser the distance (or projection error) of a particular feature from the axis, the better it is represented along that axis. In figure 15, the distance between x_i and $z_{i,1}$ is the projection error for x_i on direction 1. Eigenvalues are the scaling factors that tell us the amount of stretch or compression that occurred during the linear transformation.

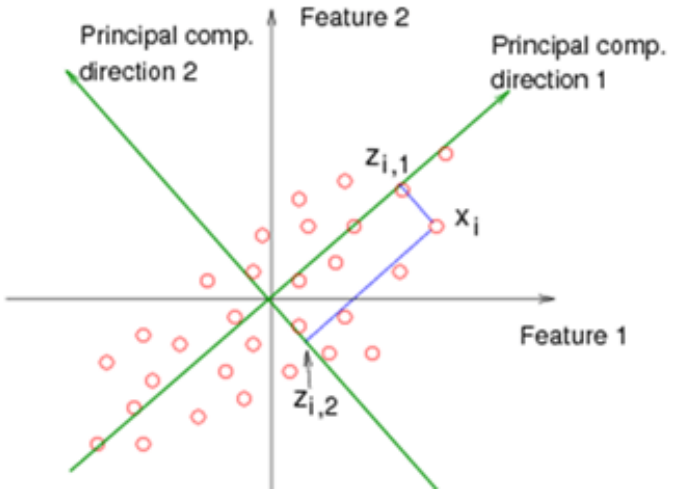


Figure 4: Principal Component Analysis (Source: onlinecourses.science.psu.edu)

3 Chapter: Literature Review

There has been a lot of work on gesture recognition incorporating the use of different types of sensors and the creation of recognition models for several real life and virtual applications. The works presenting these studies cover both IMU sensor and vision-based recognition systems. Some of these works are on full body gestures whereas some are focused on the upper body or even only on the arms. With the advent of faster and convenient mobile technologies, gesture recognition studies have also moved into portable mobile interfaces. No matter what, all of these works either use a single sensor based system or a multiple sensor based system.

uWave is a gesture recognition system that uses gesture-based interactions using single three-axis accelerometer [41]. It required a single training sample for each pattern and allows users to define their personal gestures. As the paper claims, it achieves 98.6% accuracy. The paper also presents application of uWave in gesture-based user authentication and interaction with 3D mobile user interfaces.

An automatic hand gesture recognition system has also been developed for use in Augmented Reality [12]. It is able to differentiate between static and dynamic gestures. The system uses infrared sensor to track the gestures. It tracks infrared targets mounted at the user's thumb and index finger to retrieve positional and orientation information of each finger. The authors conclude that in an Augmented Reality, the tasks executed by interaction with their gesture recognition system are faster than using conventional mouse/keyboard.

Zhu *et al* [9] created a smart assisted living (SAIL) system that can aid the elderly and the disabled using a human-robot interaction (HRI) technology. They studied hand gesture

recognition and daily activity recognition. A neural network was used for gesture spotting and a hierarchical hidden Markov model was used for context-based recognition while implementing the hand gesture recognition system. For the daily activity recognition, they developed a multi-sensor fusion system. The system uses wearable body sensors. A similar gesture spotting system was introduced in [42].

Vision-based gesture recognition has been on the rise since we started developing better cameras, image and video compression technologies, in consort with faster processors and GPUs. These systems cover several application areas such as surveillance, detection, control and other analyses of captured motion data. A wide array of vision-based approaches exist and the reader can explore [3] and [13] for further details.

Using a multiple sensor based approach, Lementec *et al* [43] presented a gesture recognition algorithm using Euler angles. Their work is a part of a control system for an Unmanned Aerial Vehicle (UAV).

Gesture recognition using virtual reality interfaces became more prominent with the invention of systems like Microsoft's Kinect, Nintendo Wii, Oculus Rift etc. In [11], a hidden Markov model based training and recognition algorithm was used on data collected from a Wii controller to develop a gesture recognition system.

Most of the studies described above were either not applied to full-body gesture recognition, or did not use wearable IMU sensors but used a vision-based approach to do so. Some full-body recognition or interaction systems can be found in the current literature, such as ALIVE [44]. A couple of view-invariant full-body gesture recognition systems are described in [45] and [46], which are still vision-based systems. The current state-of-the-art is therefore lacking in wearable sensor-based systems that can recognize full-body

gestures. Although we are only doing an upper-body gesture recognition, the outcome from our study will be the foundation to modelling a wearable sensor-based full-body gesture recognition system. Instead of using five IMU sensors, we would like to incorporate at least ten sensors for such a system.

4 Chapter: Experimental Methodology

4.1 Physical Setup

The participants were asked to strap 5 small plastic boxes on their body - 2 on both the arms and 1 on the upper abdomen (figure 4(b) – positions Ch. 13, 15, 16, 17 and 18). Each box had a sensor inside that was powered by a low voltage (1.2 Volts) battery (figure 4(a)).

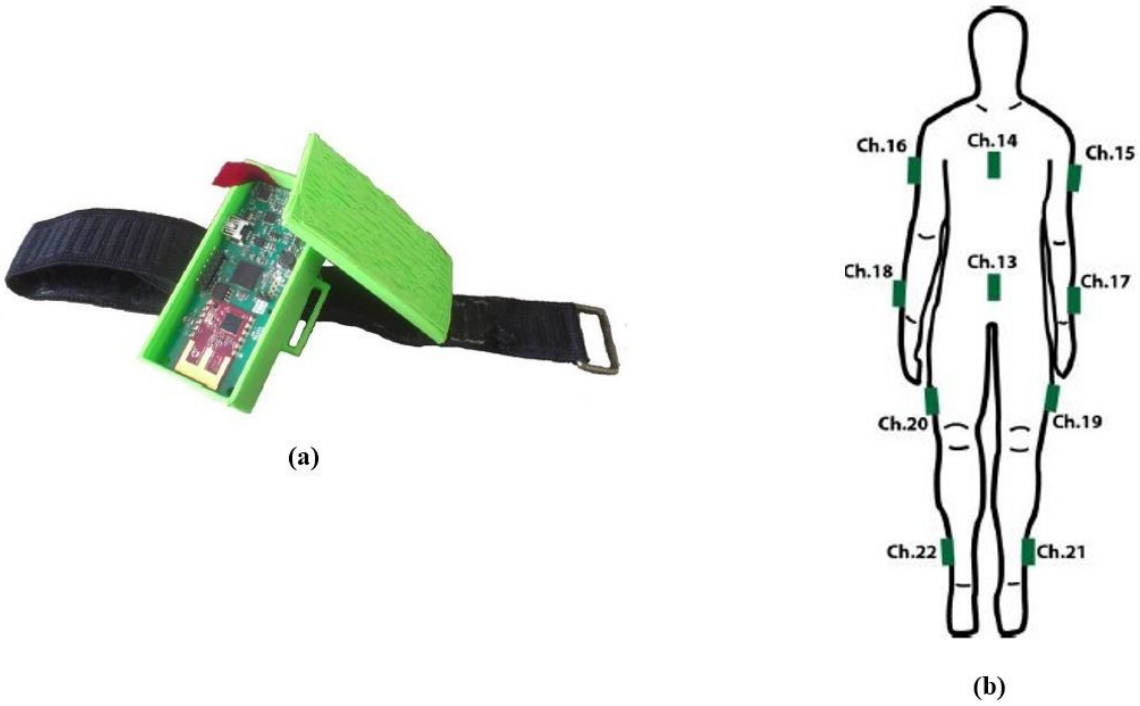


Figure 5: (a) Sensor inside its 3D printed case with Velcro strap, (b) placement of sensors on the body [7]

4.1.1 The Sensors

We used the Motion Sensing Demo Board from Microchip [47] to capture gesture data. This device is composed of a PIC24 microcontroller with a connected InvenSense MPU-6050 chip [48]. The 6050 chip contains both a 3-axis accelerometer and a 3-axis gyroscope. The demo board has a separate embedded magnetometer. The device (sensor) is powered

by a single 1.2-1.5 volts AAA battery. It transmits data wirelessly to an accompanying receiver which is a ZENA 2.4 GHz radio transceiver. It is plugged into a standard USB port of the main computer. Each sensor uses its own wireless frequency to transmit data so that their individual signals remain separated from each other. The data transmission frequency of the sensors is 110Hz.

4.1.2 Sensor Orientation

Each sensor outputs its current orientation in the form of a quaternion, about which we will discuss in the next section. Upon startup, the sensor generates its own global reference frame such that the positive z-axis is oriented vertically upwards and the x and y axes are determined by the device's orientation (figure 5) [7]. After that, the MPU on each sensor takes readings from the accelerometer and the gyroscope and calculates and estimate of the current orientation of the sensor relative to those initial axes. This process is called sensor fusion. InvenSense has not provided documentation for the exact implementation of this algorithm in their sensors. However, anyone who wishes to read further about the topic can explore [49], [50] and [51].

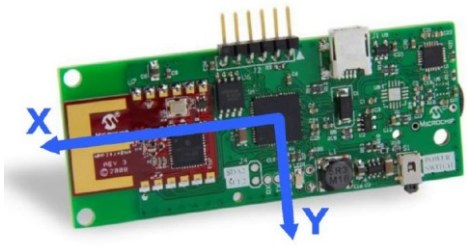


Figure 6: Direction of the sensor's global x and y axes [7]

4.1.3 Gestures

We decided to keep the selection of our gesture set as consistent as possible with the

previous study. As mentioned in section 1.4, we used six gestures: Jab, Uppercut, Throw, Lift, Block and Sway (figure 6). We kept four gestures from the previous study [7] and changed two - Jets and Asgard to Lift and Sway respectively. The reason behind this change was the difference of context between this study and the previous one. While our previous study selected gestures within the context of its robot-based action game [7], the rationale behind our selection was to use gestures that would be simpler for anyone to understand and perform. On top of this, we also wanted to have a common set of gestures with the previous study for a meaningful comparison of results. Jab, Uppercut and Block are combat gestures whereas Throw, Lift and Sway can be related to our daily lives in different ways.

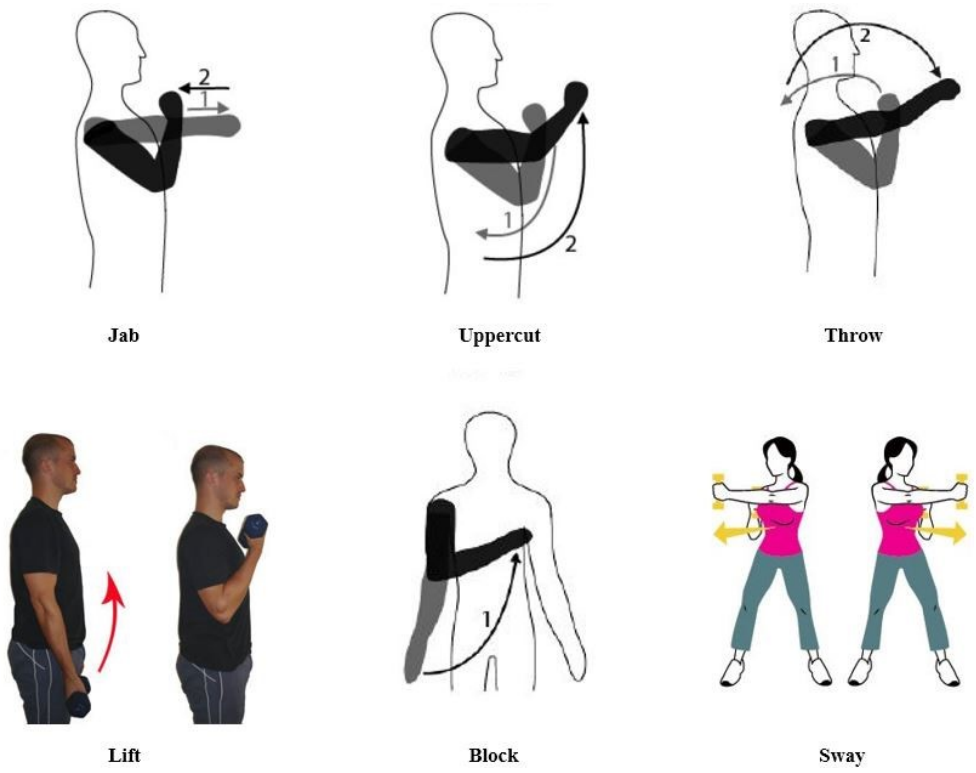


Figure 7: Gestures performed during each study

Even if these gestures are not micro gestures, e.g. pushing a button with the fingertip or controlling the joystick of the wheelchair using fingers, creating a baseline for recognizing

large gestures will pave the way in recognizing smaller or micro gestures in future.

4.1.4 The Dataset

We collected anonymous data from 11 participants who volunteered for the study, 4 females and 7 males. Every participant was shown the pictures in figure 6 one by one and was also shown a demo of each gesture before s/he was asked to perform it. We collected around 20 samples per gesture from each participant.

We consider each back and forth movement that completes a gesture (from starting the gesture to putting the limb on rest) to be an instance. Each instance of a gesture consists of several data points (coordinates) in quaternion form. We collected multiple instances of a gesture for every sample. Further explanation with exact numeric details are given below.

For each sample, the participants were asked to perform the respective gesture for about 4.5 – 5.5 seconds using one of their hands while keeping the other hand still. Thus, we were able to collect about 500-600 coordinates per sample at a sampling frequency of 110Hz. They used their left and right hands alternatively for each sample which yielded in 50% data from the left hands and 50% from the right hands (not applicable for Sway). On an average, they performed 5 instances of a gesture continuously within the 4.5 – 5.5-second timeframe per sample. Thus, each of the five sensors in our setup collected about 100 instances (20 samples * 5 instances/sample) of every gesture per participant. This yielded in a total of about 600 instances/participant which is twice the amount of the instances performed during the previous study. In addition, we used five sensors simultaneously instead of one to gather data which increased the amount of data collected to another five folds. To summarize, we have about 11 times the data (in terms of volume) as compared to the volume of the data gathered during the previous study. More details on the dataset

e.g. partitioning, size of different subsets are mentioned in the 'Data Analysis' section.

While these numbers reflect the ideal scenario, issues like missing values, participants getting fatigued contributed to the discrepancy in the numbers in the actual dataset. For example, we were able to take 20 samples for only 3 of the participants while collecting data for Sway. These are explained in more details in the following section.

4.1.5 Issues and Limitations

There were several issues that we had to deal with while performing the participant studies. We were unable to collect the same amount of data for Sway because most of the participants became fatigued from performing previous gestures. In some cases while collecting data for Sway, we had to restart one or multiple sensors by halting the procedure because the batteries were low or the reset button on the sensor(s) got automatically pressed inside their 3D printed case(s). We used only fully-charged batteries to run the sensors and replaced each one of those halfway through the study to ensure uninterrupted data collection. However, circumstances such as the one mentioned above were unavoidable in a few cases.

The sensors themselves also suffer from a drift issue which was covered in detail in the previous study [7]. Sensor drift causes the values of a continuous stream of data to deviate from their initial value even if the sensor were kept still and untouched. We found out that keeping the sensors still for 13 seconds after installing the batteries stabilizes the magnetometer inside them. This, in turn, stabilizes the sensor output.

Last but not the least, it is sometimes difficult to detect shut down(s) of one or more sensors while supervising the study. This incident occurred during one of the studies when two of the sensors went out during the performance of Lift and Block. We were able to detect this

while analyzing the dataset and discarded that portion of data.

4.2 Data Analysis

We used standard experimental methodology for a Pattern Classification or Machine Learning system to analyze our data. The data collection process has been discussed in section 3.1.4. In this section, we will explain how we built and implemented our classification system, and finally we will discuss the results.

4.2.1 Details of Implementation (Tools and Languages)

We used the following tools and programming languages to implement different parts of our experiment:

- Noise Removal Filter: Python 2.7.9
- Data Partitioning and Feature Extraction: Python 2.7.9, Microsoft Excel
- Data Visualization and Analysis: Weka 3.7 (Java-based Machine Learning Tool)
- Dataset Format: .txt (raw data), .csv and .arff (processed data)

These steps are described in details later in this thesis.

4.2.2 Volume and Quality of Data

The table below shows the amount of data that we collected from 11 participants:

Table 1: Number of Samples (5 instances/sample) in the Gesture Dataset

Gestures Performed	Left Hand Samples (LH)	Right Hand Samples (RH)	Combined (LH + RH)
Jab	102	92	194
Uppercut	105	96	201
Throw	105	96	201
Lift	95	86	181

Gestures Performed	Left Hand Samples (LH)	Right Hand Samples (RH)	Combined (LH + RH)
Block	95	86	181
Sway	66	56	122
Total # of samples:	**568**	**512**	**1080**

We have a main dataset which we call the 'Raw Dataset'. Two other datasets that stemmed from this dataset are the 'Euler Angles Dataset' and the 'Feature Dataset'. The Raw Dataset has a total of 2,959,765 coordinates in quaternion form (1 coordinate = 4 quaternion elements) per sample. This dataset also have the timestamps for each coordinate. The Euler Angle Dataset is derived from the Raw Dataset using quaternion to Euler angle conversion methods which means it also has the same number of coordinates but in Euler Angles (1 coordinate = 3 Euler angles). A breakdown of the number of coordinates collected from every participant is listed below:

Table 2: Volume of dataset

Participant ID	Number of Coordinates (in raw dataset)
1	346,673
2	313,996
3	303,172
4	314,637
5	272,907
6	163,052
7	322,276
8	308,721
9	182,250
10	327,265
11	329,200
Total	**2,959,765**

From the table above, we can say that at this stage, we have almost 3 million coordinates or 12 million quaternion elements spread among 6 different gestures for 11 different participants. We have two different sets of these coordinates: one in the condensed quaternion form (shown in table 2) and the other one in Euler angle form. Which means we have a total of 6 million coordinate. From these coordinates, we have extracted 124,200 features points, in the form of 115 identical features for every sample in the dataset. The size of all the datasets combined is about 1.3 Gigabytes.

With datasets having large volume, assessing their quality or integrity before analyzing them is of paramount importance to any Pattern Classification experiment. Here, we use the term 'quality' to refer to the amount of missing values in the datasets and the consistency of the data such as class distribution. Issues like sensor drift and sensor shut down have already been explained above in section 3.1.5.

Missing Values: We found a few missing values in most of the subsets (individual participant studies). Although negligible in number, our investigation revealed that a sensor or a couple of sensors suffered from noise due to internal technical problems that yielded quaternion values such that it defied equation 10 (magnitude greater than 1). We discovered this while calculating the Angular Velocity feature during the Feature Extraction process (explained in section 3.2.5). To replace each missing value, we considered a tiny portion of the gesture (by visualizing the graph of that gesture sample) such that the value is located in the middle. Then we took an average of that portion so that the continuity of the graph remained intact. In some cases, it was impossible to take the average value(s) to replace a long array of missing or zero values. This situation occurred mainly due to hardware fault and we had to discard the affected samples. It is explained in details below.

Consistency in Dataset: We tried to keep the class distributions and sample counts in the dataset consistent in every subset. However, from table 1, we can see that there are three discrepancies in the number of samples

1. The right hand gesture set has 56 less samples than that of the left hand gestures.
2. Number of sample for Sway is lesser than that of the others.
3. Lift and Block has lesser number of samples.

The first one occurred due to a small bug in the code that adds appropriate header column and separates right and left hand samples. The first row (or sample) in each subset of raw data is a right hand sample, followed by a left hand sample and it is thereby arranged in an alternating manner. While writing header columns, it overwrote the first row of each file, consequently deleting one right hand sample per gesture for every participant. As it is only 5% of the whole data, the glitch is uniformly spread throughout the entire dataset and has no noticeable effect on the results (which we will be able to see later), we decided not to redo the initial data partitioning process due to time constraints.

On the other hand, Sway has lesser number of samples. It was the last gesture that the participants had to perform. By that time, some of them became tired. Moreover, due to hardware issues and routine battery changes during every study, it was sometimes difficult to finish everything within the 1-hour time limit. One of the participants could not perform Sway due to such time constraints. We still managed to gather 20-sample datasets for Sway from 3 of the participants. We designed our participant studies in such a way that the gestures are captured in the same order for every participant. The easiest and most unique gesture was performed last. Sway was the solution to this because it made use of all of the sensors (especially the sensor that was attached to the abdomen) as opposed to the other

ones where the abdomen sensor sat idle. This made the gesture quite different as compared to the other ones. Since it uses a unique feature, we made the assumption that the discrepancy in overall class distribution due to lesser number of samples in Sway would not pose a problem during classification. The final results proved our assumption to be true. We excluded 20 bad samples each from Lift and Block because one of the sensors shut down while one of the participants was performing these gestures.

To summarize, our dataset reflects practical situations that can arise during a standard data collection process. Participants' fatigue during performing the gestures might have had an effect on the captured data. We performed our experimental analysis keeping this issue in mind. We also made sure that the data are as consistent and clean as possible before starting the analysis.

4.2.3 Denoising: Outlier Detection and Removal

We collected our own data to test our data collection process before carrying it out for the actual participant study. I performed each gesture for 40 minutes and collected a total of 240 minutes of gesture data for testing purposes. This part was extremely important to our experiment design because it revealed the limitations of rechargeable batteries that are used to run the sensors. We used Duracell and Energizer batteries to power the sensors. The battery ratings are as follows:

- Voltage: 1.2 volts
- Type: NiMh (Nickel-Metal Hydride), AAA
- Energy: 500 (x 4*), 700 (x 4) and 800 (x 12) mAh

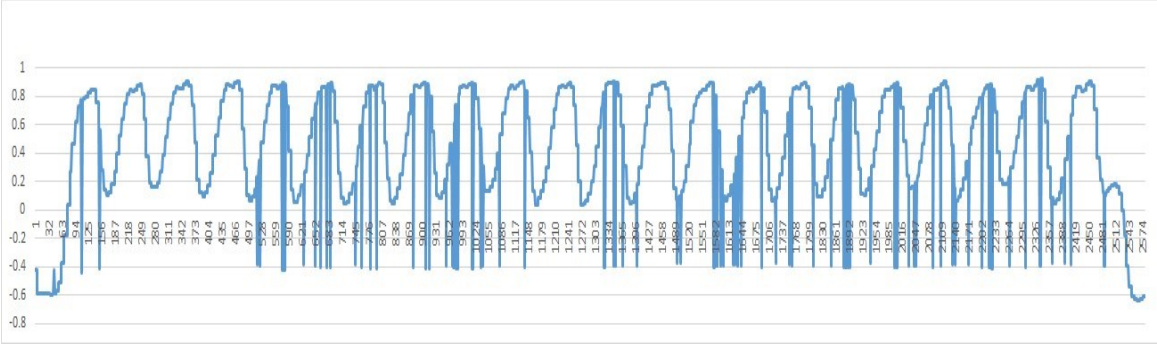
* Number of batteries used

While analyzing this dataset, we found that it contained a lot of noise or outliers (figure 8).

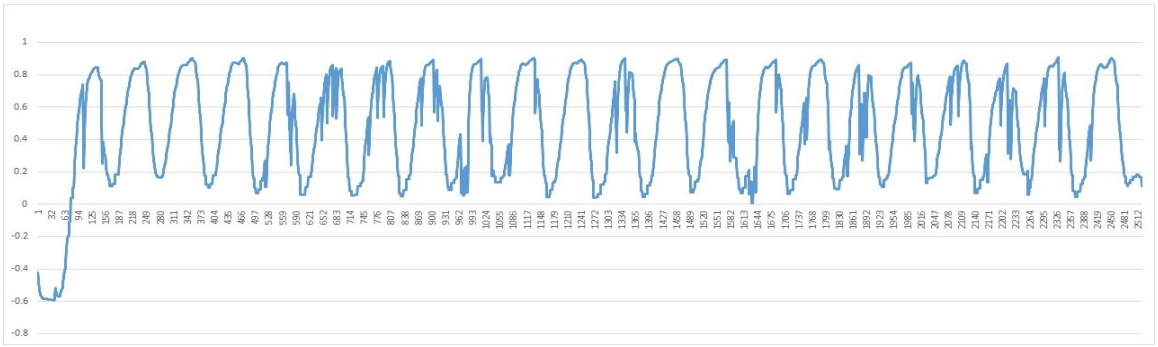
The reason was low or dead battery during the study. As the energy level of the battery started to drop, the sensor faced frequent interruptions in data transmission and the values suddenly dropped during that period. This phenomenon continued till the battery was dead and the sensor eventually shut down.

We developed a **Two-Step Noise Removal Filter** to reduce these types of noises in the data. It is based on the idea that uninterrupted gestures are usually continuous streams of data. Anomalies in the continuity of the data stream can thus be interpreted as noise or outliers. A brief overview of how the filter removes noise in 3 steps is given below:

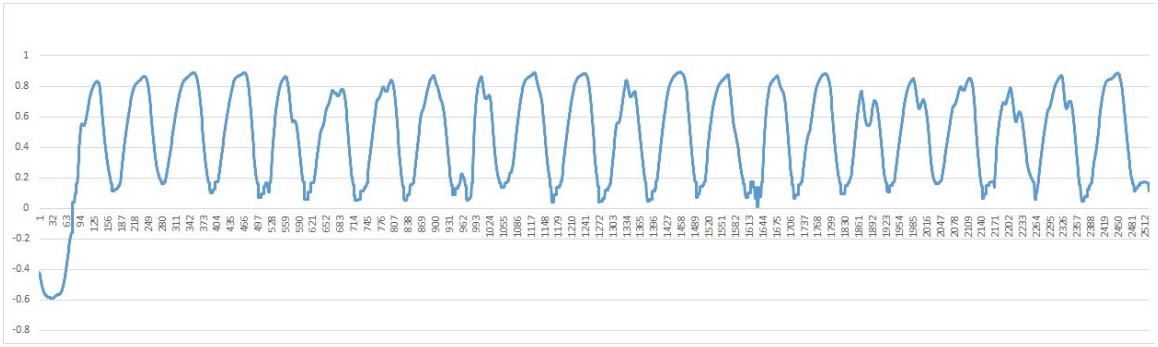
1. Calculate Euclidean distance between a reference point and every consecutive point within a window of finite width.
2. If the distance is greater than a certain threshold, perform linear interpolation within the window to smoothen the data (figure 8) and translate the window forward until it reaches the end of the stream.
3. Repeat step 1 and 2 but calculate first derivative instead of Euclidean distance within the same window. This will remove noises that remain undetected during the first two steps and finally result in a smoothened dataset (figure 9).



(a)



(b)



(c)

Figure 8: (a) Noisy data (b) Partial noise removal (c) Noise mostly removed after derivative filtering

Fortunately, changing the batteries half-way through the studies and replacing them with newly charged ones resulted in datasets with no such noise or outliers. As a result, we did not have to use this filter for our actual experiment.

4.2.4 Data Partitioning

Similar to our previous study, we organized the data into two categories:

1. Generalized Gesture Recognition

The generalized dataset may include test set data from any participant. Here, we are not interested in the individual from whom the data are coming but only interested in the gestures. This rule applies to any of the training, cross-validation or test set under this category. In short, this dataset can be taken 'out of the box' and used to test on any new data for the 6 gestures we included in this study.

2. User Specific Gesture Recognition

Every user specific dataset comprises test set that contains data from a specific individual within the participants. We made 3 different test cases and none of them contains any common sample. For example, case P1 has a test set that contains data from only participant 1 but case P7 has a test set that includes data from participant 7 only. This method tests whether our model is capable of recognizing a particular person's gestures as opposed to recognizing any gesture in general, which is a probable reflection of a real life gesture recognition scenario. During the selection process of our test sets, we tried to maintain fairness by randomizing the sequence of the participant datasets before selecting the three mentioned above. A leave-one-subject-out testing would have been the best option here. However, due to the complex structure of the dataset, we decided not to perform a leave-one-subject-out testing because of time constraints.

Both the categories have separate datasets corresponding to left hand gestures, right hand gestures and a combined dataset that includes data from both of these sets. Each dataset is divided into three parts: training, cross-validation and test sets. The training sets were used to train the classifiers. Cross-validation sets were used as pseudo test sets not only to tune classifier parameters but also to get insight on how to take further steps on improving recognition accuracy. Every corresponding test set was used only once to test the performance of the classifiers. The test sets were kept independent, i.e. no data from the test set was used as part of the training process at any point during the experiment.

Data proportions are 60% for training, 20% for cross-validation and 20% for testing. To ensure that our experimental methodology generalizes well for different types of partitions, we chose not to have a cross-validation set for the combined case. Instead, we decided to

use same tuned parameters obtained from the validation tests for the two other datasets (left hand and right hand). Two different partitions were made for the combined case using the same data with proportions being 70%-30% and 60%-40% respectively.

Data are usually partitioned in this manner to ensure proper parameter selection by testing different combination of classifier parameters on different training – cross-validation sets. For a certain combination, if the classifier's performance on the cross-validation set is consistent, then it is deemed appropriate for testing. Otherwise, another combination is tested on the validation set. Figure 9a below shows the data partition hierarchy for the generalized case and figure 9b shows the same for the user specific case.

We used Weka's 'Supervised Resampling' filter to create partitions for the generalized case. This filter randomly reorders the dataset and then divides it into proportions set by the user while maintaining proper class distribution in the subsample(s). We made sure that the resampling was done without replacement i.e. no sample was repeated twice. For the user specific case, we created partitions manually according to figure 9 and performed randomization before training the classifiers.

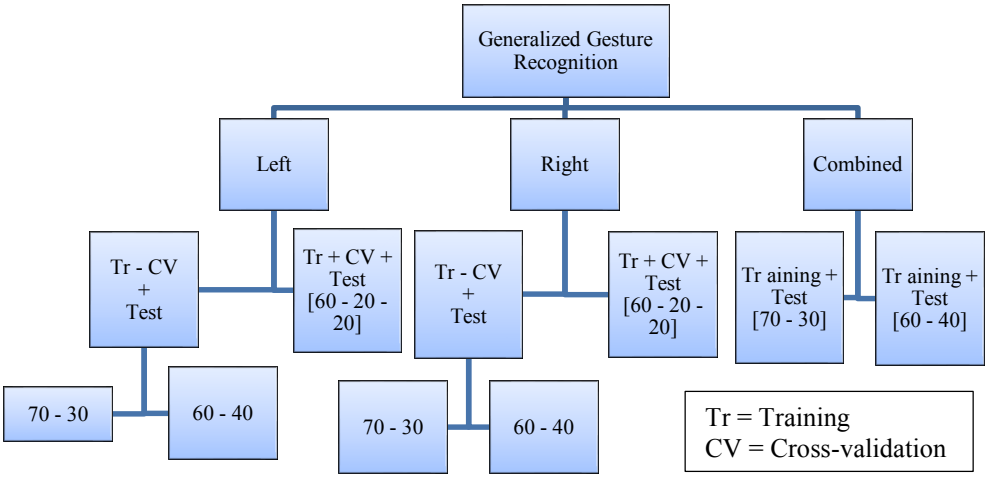


Figure 9a: Datasets for Generalized Gesture Recognition

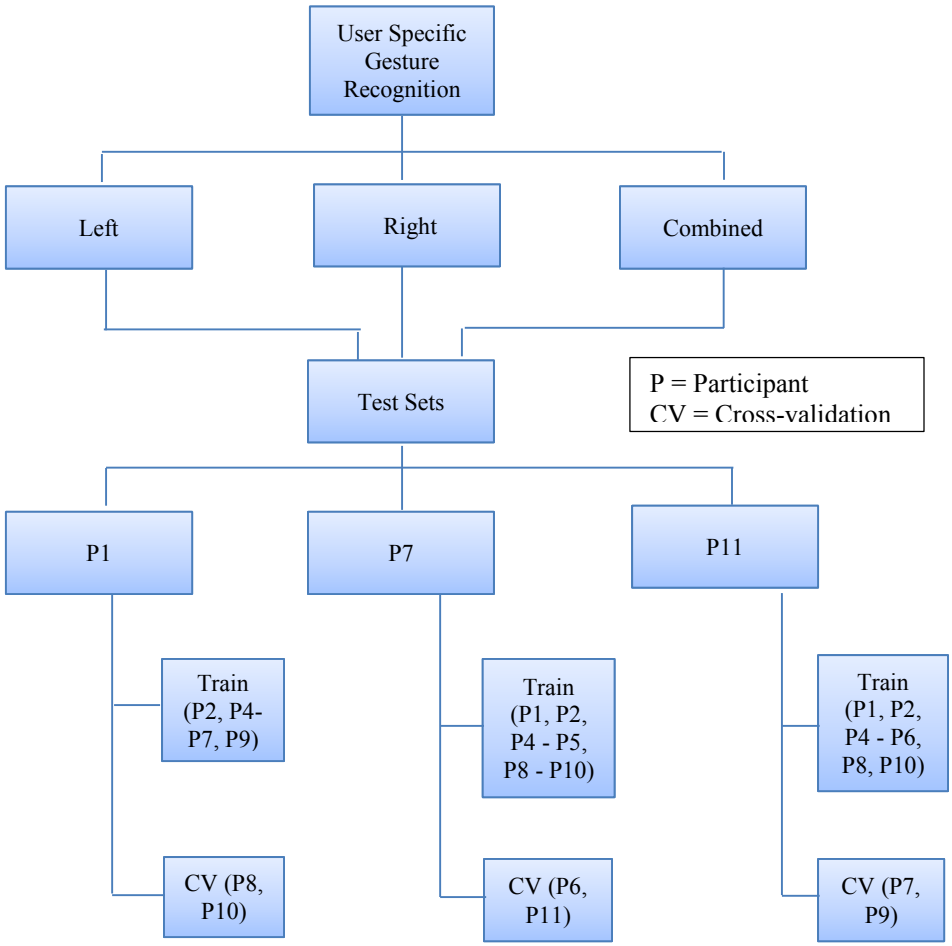


Figure 9b: Datasets for User Specific Gesture Recognition

In the user specific case, we excluded data from participant 3 because two of the gestures for this participant had bad data due to a sensor fault. However, we did not totally exclude this participant from the generalized case. As the generalized case is not participant dependent, we kept the good data for the other 4 gestures performed by participant 3.

4.2.5 Feature Extraction

We calculated the following sets of features from our dataset:

- Variance
- Range

- Velocity
- Angular Velocity
- Covariance

Feature Labelling: Every quaternion element from every sensor was considered as a source of feature. The sensors were labelled as Sensor 15, Sensor 16, Sensor 17, Sensor 18 and Senor 19. As the sensors output data in the form of four quaternion elements (q_r, q_x, q_y, q_z), we separated the respective columns and used each element to calculate one feature. For example, a feature (single column in the feature dataset) labelled as 'var_15_qr' refers to variances calculated using the q_r values from each sample of Sensor 15 for a particular gesture performed by an individual. We coined these types of features as 'Single Sensor-based Features'. On the other hand, a feature labelled as 'cov_15_16_qx' refers to the covariance of Sensor 15 and Sensor 16's output for the quaternion element q_x. Thus we have two different types of feature sets: Single Sensor-based Features and Multiple Sensor-based Features. In total, we have 115 features. Each sensor contributed to the single sensor-based features individually. The multiple sensor-based features constitute all possible combinations of the five sensors and four quaternion elements.

4.2.5.1 Single Senor-based Features

Variance, Range, Velocity and Angular Velocity are calculated using data from single sensor at a time, adhering to the first example given above. Details of the feature calculations are shown below while a complete list of features is shown in Appendix A.

Variance: It is the average of the squared deviations from the mean. In simpler terms, it measures the spread of data points around the mean by calculating how far each number in the set is from the mean. It is calculated as:

$$variance = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n} \quad \text{..... (13)}$$

Here, x_i = value of data point at the i-th index of the sample

μ = mean

n = number of data points in the sample

Variance not only provides intuitive information about a dataset but is computationally faster to calculate than features such as velocity or angular velocity as well. We have 20 variance features for every gesture, each corresponding to a different sensor and a specific quaternion element. A few examples would be Var_15_qr, Var_16_qx, Var_19_qz etc.

Range: It is the difference between the maximum and the minimum value of a single sample. It is the easiest way to measure the spread of the data. Although it is a crude measure of variability, we chose to use it as one of the features because it would be able to differentiate between short and long gestures such as Jab and Uppercut.

$$range = x_{max} - x_{min} \quad \text{..... (14)}$$

Here, x_{max} = maximum value in the sample

x_{min} = minimum value in the sample

Like variance, we have 20 range features for every gesture, each corresponding to a different sensor and a specific quaternion element. A few examples would be Range_17_qy, Range_15_qx, Range_19_qz etc.

Velocity: By definition, velocity is distance travelled over time towards a specific direction i.e. the speed of an object towards the direction it is travelling. We considered this as an important feature because every gesture is unique and therefore should have varying degrees of velocity which may contribute to classification. Moreover, it is expected that different participants will perform gestures at different velocities. We calculated distance

summing over the Euclidean distances of each consecutive data points in every sample. Likewise, velocity was calculated in the following manner:

$$velocity = \frac{total\ distance\ covered}{total\ time\ spent\ to\ cover\ the\ distance} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (15)$$

$$= \frac{\sum_{i=1}^{n} Euclidean\ distance\ (x_{i+1},\ x_i)}{time} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (16)$$

There are two ways to calculate time from our dataset. One way is to use range of the timestamps ($timestamp_{max} - timestamp_{min}$) to calculate the elapsed time for each sample. Another way is to divide the number of data points by sensor frequency. Both are essentially the same. We used the latter one to save computation time.

$$(elapsed)\ time = \frac{number\ of\ datapoints}{sensor\ frequency} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (17)$$

In equations 16 and 17, x_i = value of data point at the i-th index of the sample

n = number of data points in the sample

Sensor frequency = 110 Hz

Examples of velocity in the dataset are Vel_15_qz, Vel_18_qr etc. There are 20 velocity features in the feature dataset.

Angular Velocity: It is the rate of change of angular displacement of an object about its axis of rotation [52]. In other words, it is the rate of change of angular positions of a rotating body. This feature gives us positional information of the active limbs in 3-d space. The rationale behind using this as one of features is similar to that of using velocity. We can calculate angular velocity from quaternions in two steps:

1. **Convert Quaternions to Euler Angles**

Euler angles are three angles that show a particular sequence of three rotations of a rigid body about particular reference frame axes. They can describe the orientation of a rigid

body in 3-dimensional Euclidean space. From the different parameterizations that are possible to convert quaternions to Euler angles [53], we chose to use:

$$\alpha = atan(\frac{2(q_w q_x + q_y q_z)}{1-2({q_x}^2 + {q_y}^2)}) \quad \text{..... (18)}$$

$$\beta = \text{asin}(2(q_w q_y - q_x q_z)) \quad \text{..... (19)}$$

$$\gamma = \text{atan}(\frac{2(q_w q_z + q_x q_y)}{1-2({q_y}^2 + {q_z}^2)}) \quad \text{..... (20)}$$

Where,

$$-\pi < \alpha \leq \pi$$

$$-\frac{\pi}{2} < \beta \leq \frac{\pi}{2}$$

$$-\pi < \gamma \leq \pi$$

q_w, q_x, q_y, q_z constitute a unit quaternion q.

2. Calculate Angular Velocity Vectors

From the angles calculated in equation 18, 19 and 20, we can calculate the angular velocity vectors Precession, Nutation and Spin as the following:

$$precession = \dot{\alpha} \sin(\gamma) \sin(\beta) + \dot{\beta} \cos(\gamma) \quad \text{..... (21)}$$

$$nutation = \dot{\alpha} \cos(\gamma) \sin(\beta) - \dot{\beta} \sin(\gamma) \quad \text{..... (22)}$$

$$spin = \dot{\alpha} \cos(\beta) + \dot{\gamma} \quad \text{..... (23)}$$

Here, similar to equation 16,

$$\dot{\alpha} = precession\ velocity = \frac{\sum_{i=1}^{n} Euclidean\ distance\ (\alpha_{i+1},\ \alpha_i)}{time} \quad \text{..... (24)}$$

$$\dot{\beta} = nutation\ velocity = \frac{\sum_{i=1}^{n} Euclidean\ distance\ (\beta_{i+1},\ \beta_i)}{time} \quad \text{..... (25)}$$

$$\dot{\gamma} = spin\ velocity = \frac{\sum_{i=1}^{n} Euclidean\ distance\ (\gamma_{i+1},\ \gamma_i)}{time} \quad \text{..... (26)}$$

$$(elapsed)\ time = \frac{number\ of\ datapoints}{sensor\ frequency_{quat}} \quad \text{..... (27)}$$

In equation 27,

$sensor\ frequency_{quat}$ = data transmission frequency after conversion to Euler angles (3 angles instead of 4 quaternion elements)

= 82.5 Hz

These features are labelled as Precession_15, Nutation_16, Spin_19 etc. Our feature dataset has 15 Angular Velocity features.

4.2.5.2 Multiple Sensor-based Features

Covariance is the only feature that uses data from multiple (two) sensors in every feature. As the name suggests, it calculates the covariance between any two sensors for a particular quaternion element. In this case, it indicates the level to which the output of two sensors vary together. We can calculate the covariance of Sensor 15 and Sensor 16 for q_w as the following:

$$Cov\_15\_16\_qw = \frac{\sum_{i=1}^{n}(x_{15_i} - \mu_{x_{15}})(x_{16_i} - \mu_{x_{16}})}{n} \quad \text{..... (27)}$$

Where, x_{15_i} = value of data point at the i-th index of the sample for Sensor 15

x_{16_i} = value of data point at the i-th index of the same sample for Sensor 16

$\mu_{x_{15}}$ = mean of the sample for Sensor 15

$\mu_{x_{16}}$ = mean of the sample for Sensor 16

In total, we have 40 different combinations of Covariance features.

4.2.5.3 Visualization

Figure 9 shows examples of different features taken from the left hand feature dataset for Participant 1. Looking at the variations in these samples, we can see that sensor 15 and 16 do not have substantial contribution to the spread of the data whereas features involving

sensor 17 and 18 have a broader variation. This is expected because left hand gestures involved movements of sensors 17 and 18 while sensors 15 and 16 placed on the right hand and sensor 19 placed on the abdomen remained idle during that time.

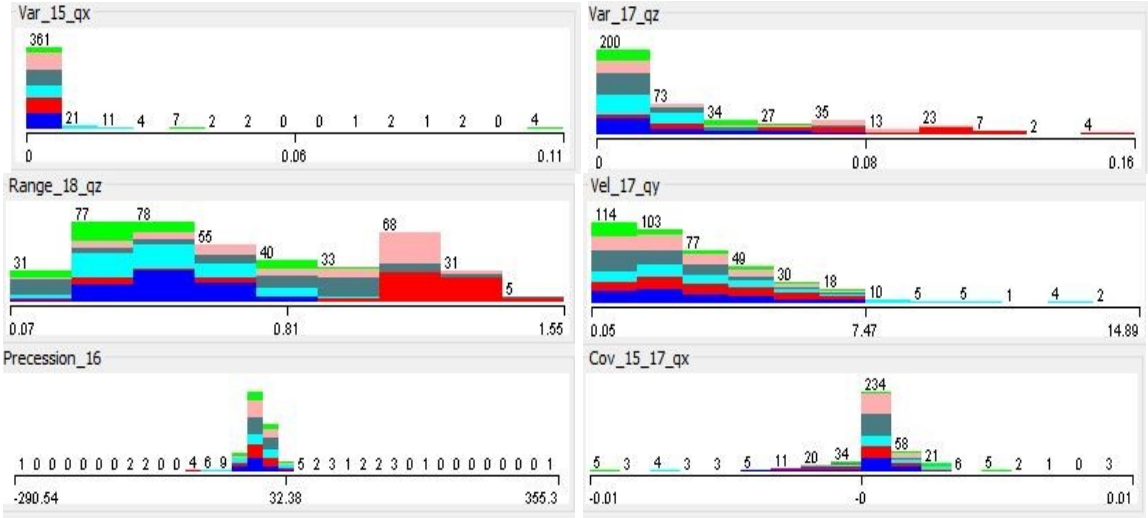
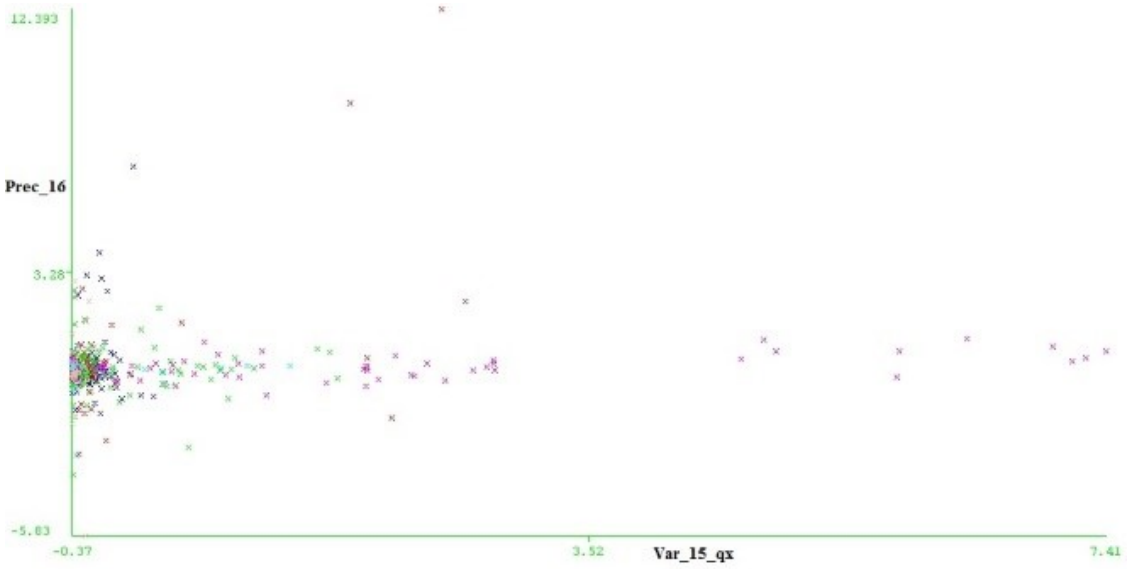


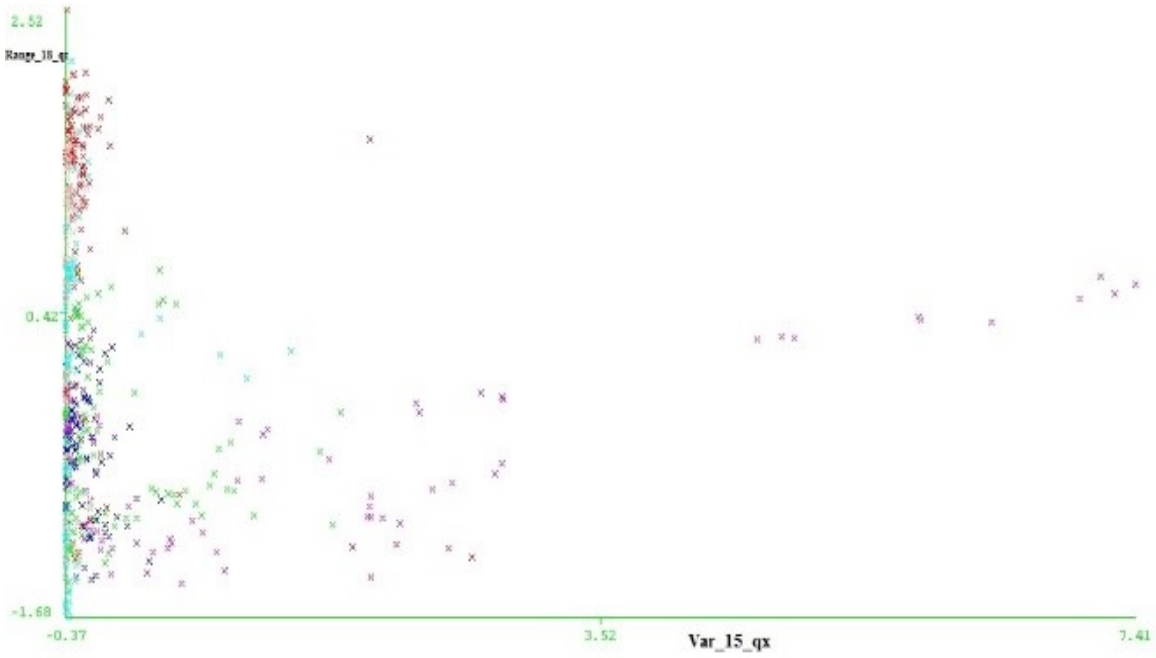
Figure 10: Different features showing variation in data

The different colour bars in figure 10 indicate different classes or gestures.

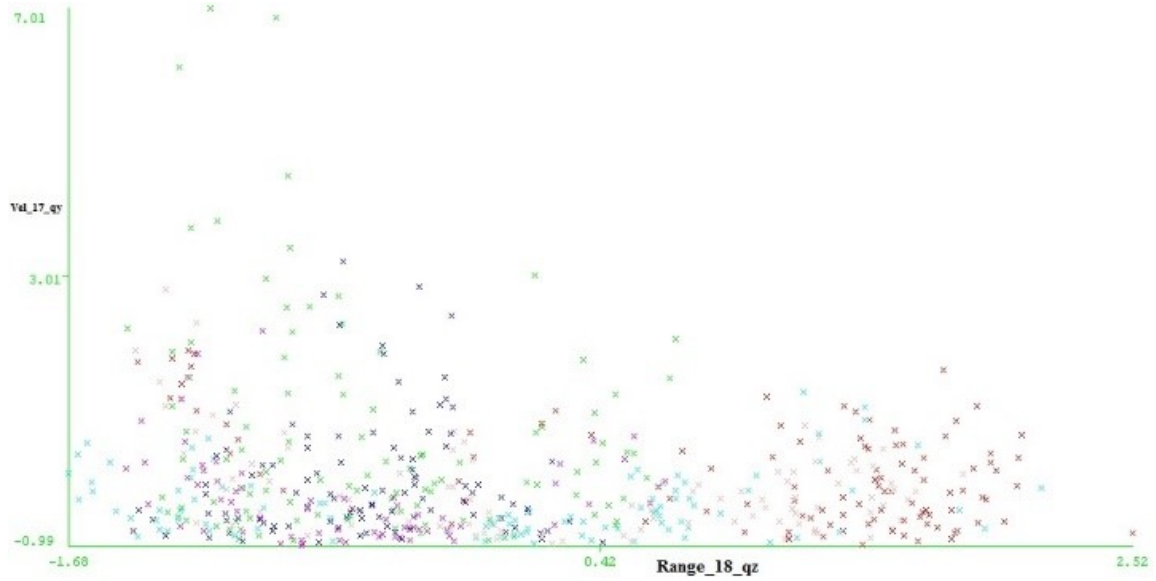
Let us look at the plots of 3 different sets of feature pairs from the figure above to examine if the explanation holds:



(a)



(b)



(c)

Figure 11: Data spread between (a) Var_15_qx and Prec_16 (b) 31. Var_15_qx and Range_18_qz (c) Range_18_qz and Vel_17_qy

Figure 10 shows plots between two presumably bad features (fig. 10(a)) for a left hand gesture - Var_15_qx and Prec_16, one bad and one relatively good feature (fig. 10(b)) - Var_15_qx and Range_18_qz, and two relatively good features (fig. 10(c)) - Range_18_qz

and Vel_17_qy. We can see that figure 10(a) shows highly condensed data near the lower y-axis which means there is very little variation for a classifier to work with. The data in figure 10(b) does not look as condensed as that of the previous one because we have one good feature there. We can also observe a little bit of class separation (different colours correspond to different classes). However, most of it is still clamped around the y-axis which is still not the best indicator of classification. In figure 10(c), the data clearly has a nice spread along both the x and y-axis. Overlaps among the feature points are minimal and class separation is clearly visible. This proves our assumption from initial visualization (figure 9) that during classification, features from the sensors that were most active during a particular gesture would be the best inputs for our classifiers in terms of class separation. However, we need to find a generalized method to find the best combinations for different datasets (left, right or combined) and put less emphasis on the ones that are not useful. This is discussed in details in section 3.2.8.

4.2.6 Data Preprocessing

After creating necessary partitions, we standardized the datasets so that they have zero mean and unit variance. 'Zero mean' is obtained by simply removing the mean of the data vector (row or column vector) from every data point in that vector. 'Unit variance' is achieved by dividing each data point by the standard deviation of the corresponding vector. Standardization (also called scaling) was performed in batches such that the cross-validation and test sets have same standardized outputs as their corresponding training sets. It is important that we perform it in this way. Otherwise, performing standardization on two datasets separately will most certainly create two differently standardized sets since the mean and standard deviation is based on the input data. Then, these parameters will

differ if the datasets are different and we will lose consistency.

Standardization is also very useful for classifiers such as SVM and Neural Network. Scaling the data to fit into a smaller numeric range such as [0, 1] or [-1, +1] lets all features contribute equally to the classification process. Otherwise, features with greater numeric values will dominate over features with smaller numeric ranges and we will essentially get a biased classification. On the other hand, standardizing the inputs can make training faster because of the uniform, smaller range of the dataset. This is very beneficial in training Artificial Neural Networks as it reduces its chances of getting stuck in a local optima.

4.2.7 Parameter Selection and Initial Experiment

Now that our data is all set for training, we decided to perform a quick experiment to understand how the two classifiers, SVM (Support Vector Machine) and ANN (Artificial Neural Network) perform after being trained with scaled data. We also used the results from this step to tune the parameters of the classifiers. We ran this experiment for both the generalized and user specific case. The parameters used for the classifiers were:

SVM: cost is set to 1.0 to 3.0 with increments of 0.5, kernel is set to linear. Varying the cost parameter over 1.0 did not yield any difference in the results. Therefore, we decided to use 1.0 later in our experiment as well.

We used linear kernel because we have a large number of features. Non-linear kernels map data onto a higher dimensional space, which we do not require for our feature set.

ANN: 1 through 20 and 'a' numbers of hidden layers were tested where,

$a = \frac{\#\ of\ features + \#\ of\ classes}{2} = \frac{115+6}{2} \approx 60$, learning rate, α is set to 0.1, momentum, m is set to 0.2, epoch is set to 500, validation set size is set to 20%, validation threshold is set to 20.

Here, momentum is used to diminish the fluctuations in weight changes over consecutive iterations. Epoch is a measure of the number of times the entire training set is used to update the weights, followed by a testing of the validation set of a user specified size (20% in our case). Validation threshold dictates how many times in a row the validation set error can get worse before training is terminated. We ran the experiments for all of the hidden layer settings (16 runs) mentioned above. The corresponding training and validation set accuracies are shown in figures 12 through 15.

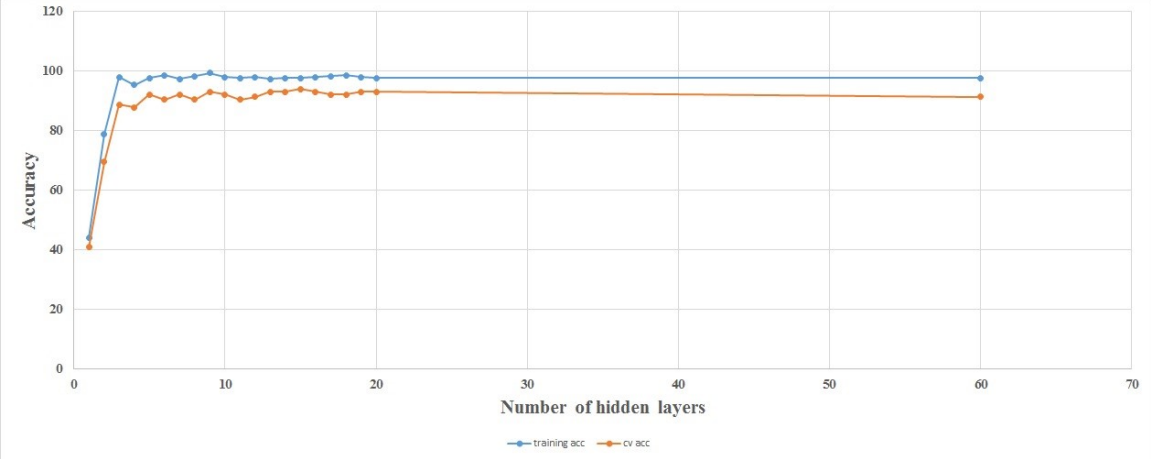


Figure 12: ANN validation set accuracies for different number of hidden layers (LH, generalized)

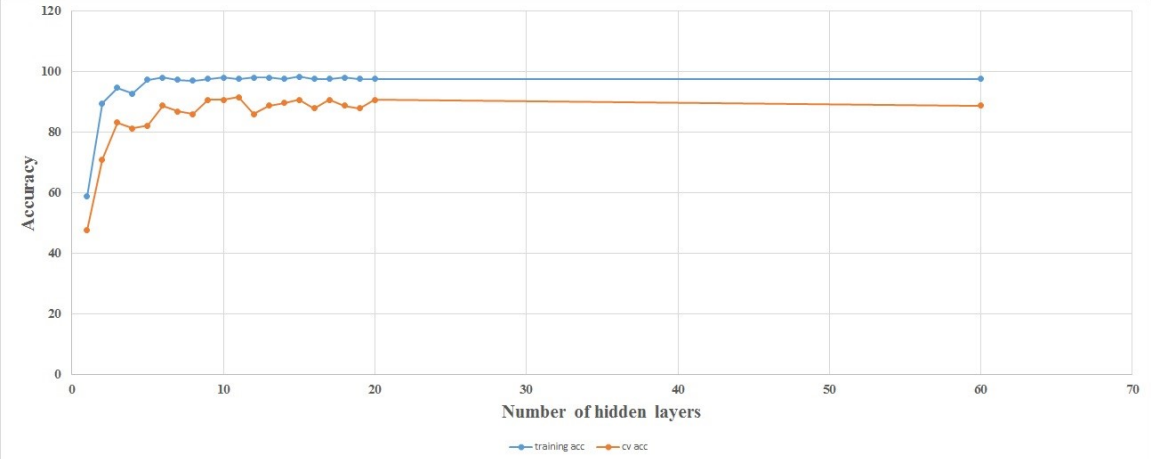


Figure 13: ANN validation set accuracies for different number of hidden layers (RH, generalized)

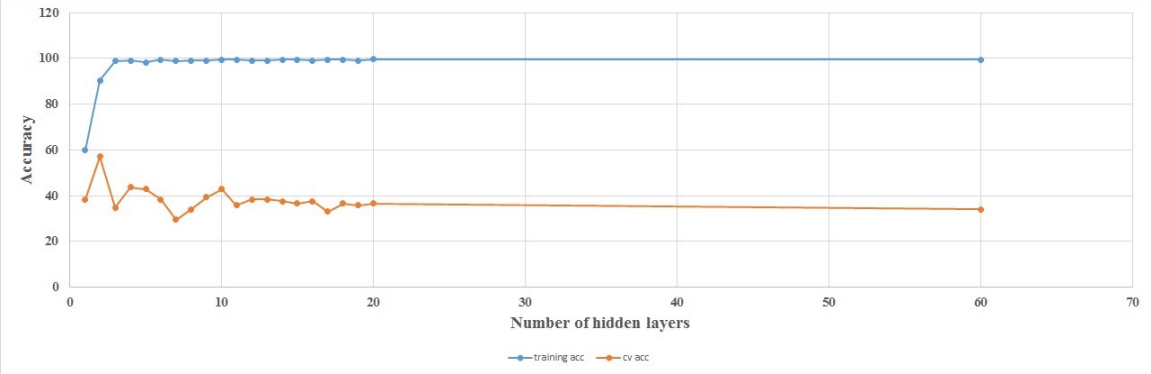


Figure 14: ANN validation set accuracies for different number of hidden layers (LH, user specific)

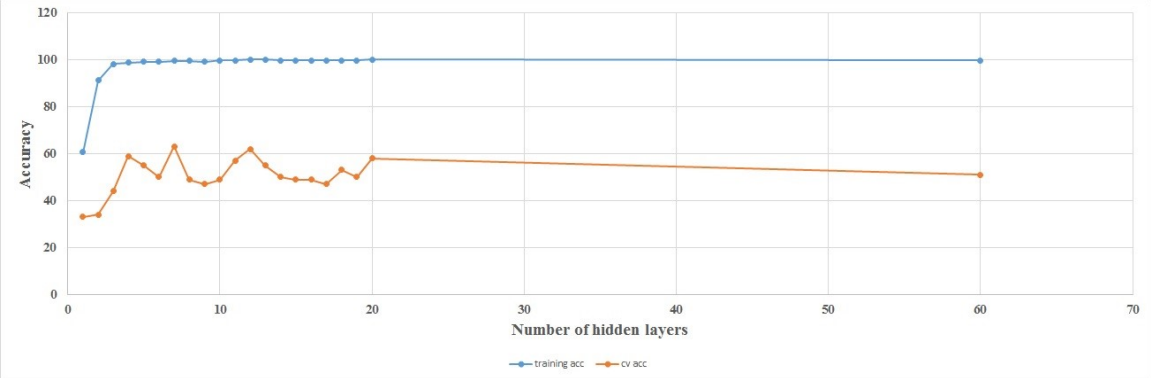


Figure 15: ANN validation set accuracies for different number of hidden layers (RH, user specific)

From the figures above, we can see that for generalized cases, the training set accuracies start to settle when the number of hidden layers hit 10. For user specific cases, it starts to settle from 3. However, validation set accuracies start to settle at around 15 hidden layers. Although they go back and forth for right hand gestures (figure 12 and 14), especially for the user specific cases, we found that 15 hidden layers yield an accuracy closer to the average value (excluding values for layer 1 and 2 because these are unstable). This is true for all other cases as well. The average accuracies over all hidden layer sizes for each case are shown in table 3. According to these results, we chose to use 15 hidden layers for every subsequent experiment. A complete table of individual accuracies is given in Appendix B.

Table 3: Accuracies of ANN for different number of hidden layers

Cases	Training set average accuracies (%)	Training accuracy with 15 layers (%)	Validation set average accuracies (%)	Validation accuracy with 15 layers (%)
Left hand, generalized	97.82	97.65	91.76	93.91
Right hand, generalized	97.33	98.36	87.95	90.65
Left hand, user specific	99.25	99.45	37.12	36.61
Right hand, user specific	99.49	99.70	52.47	49.00

Below are the results of our initial experiments:

4.2.7.1 Generalized Gesture Recognition

Training and validation set accuracies for the different datasets under this category along with classifier training time is shown below in table 4. The confusion matrices for two of these datasets in table 5 and 6 tell us where the misclassifications took place. Misclassified gesture pairs are marked in red in every confusion matrix.

Table 4: Training and Validation set accuracies for different datasets (generalized category)

Dataset	Classifier					
	SVM			ANN		
	Training Time (seconds)	Training Accuracy (%)	**Validation Accuracy (%)**	Training Time (seconds)	Training Accuracy (%)	**Validation Accuracy (%)**
Left Hand	1.76	99.71	**93.04**	12.62	97.65	**93.91**
Right Hand	1.03	100	**94.39**	12.69	98.36	**90.65**

The results above indicate that despite a few classification errors, the classifiers' accuracies on the validation sets are over 90%, which is a pretty acceptable range without having to perform any feature selection or dimensionality reduction. The few misclassifications that

were observed during this experiment are detailed below in the confusion tables:

Table 5: Confusion matrix (in %) for validation set accuracy of Left Hand (60-20-20), SVM

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**90.5**	0	9.5	0	0	0
Uppercut	0	**100**	0	0	0	0
Throw	0	0	**100**	0	0	0
Lift	0	5.2	0	**84.2**	10.6	0
Block	0	5.2	0	10.6	**84.2**	0
Sway	0	0	0	0	0	**100**

As seen in table 5, the gesture mix-ups are Jab with Throw**,** and Lift with Block and Uppercut.

Table 6: Confusion matrix (in %) for validation set accuracy of Right Hand (60-20-20), SVM

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**78.9**	0	21.1	0	0	0
Uppercut	0	**85.0**	10.0	5.0	0	0
Throw	0	0	**100**	0	0	0
Lift	0	0	0	**100**	0	0
Block	0	0	0	16.7	**83.3**	0
Sway	0	0	0	0	0	**100**

As seen in table 6, the gesture mix-ups are Jab with Throw**,** Uppercut with Throw and Lift, and Block with Lift.

Looking at the confusion tables above, we can see that the classifiers are mostly confusing Jab with Throw, Uppercut with Throw and Lift, Block with Lift and in some cases, Uppercut with Block (shown in Appendix C). It is evident that the classifiers are having difficulty in separating similar gestures, which is something one would expect from a gesture recognition problem.

4.2.7.2 User Specific Gesture Recognition

We used the test case 'P7' (please see figure 9) to examine our parameter selection and classifier accuracy for this category. The format for this experiment is the same as above. Table 7 shows training and validation set accuracies for the different datasets along with classifier training time. Misclassified gestures in the confusion matrices (tables 8 and 9) are marked in red and unrecognized gestures are marked in blue.

Table 7: Training and Validation set accuracies for different datasets (user specific category)

Dataset	Classifier					
	SVM			ANN		
	Training Time (seconds)	Training Accuracy (%)	**Validation Accuracy (%)**	Training Time (seconds)	Training Accuracy (%)	**Validation Accuracy (%)**
Left Hand	1.06	100	**38.39**	13.65	99.45	**36.61**
Right Hand	1.03	100	**58.00**	12.62	99.70	**49.00**

The results in table 7 show that the classifiers failed to generalize the data during the training phase which resulted into poor accuracies in the cross-validation phase. Two confusion tables below show the gestures (highlighted) which the classifiers found to be most difficult to classify.

Table 8: Confusion matrix (in %) for validation set accuracy of Right Hand, ANN

	Jab	**Uppercut**	**Throw**	**Lift**	**Block**	**Sway**
Jab	**0.0**	0	41.2	35.3	23.5	0
Uppercut	0	**0.0**	0	0	100	0
Throw	35.0	0	**65.0**	0	0	0
Lift	0	0	0	**50.0**	50.0	0
Block	10.0	5.2	0	35.0	**50.0**	5.0
Sway	0	0	33.3	0	0	**66.7**

As seen in table 8, the gesture mix-ups are Throw with Jab**,** Block with Lift and Sway with Throw. Two of the gestures, Jab and Uppercut, were not recognized at all by the classifiers.

Table 9: Confusion matrix (in %) for validation set accuracy of Left Hand, SVM

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**26.7**	0	60.0	6.65	6.65	0
Uppercut	55.6	**44.4**	0	0	0	0
Throw	5.55	0	**55.6**	5.55	33.3	0
Lift	0	0	0	**88.9**	11.1	0
Block	5.6	0	33.3	11.1	**50.0**	0
Sway	0	0	53.9	30.7	0	**15.4**

Table 9 shows that the classifiers confused Jab with Throw**,** Uppercut with Jab**,** Throw with Block**,** and Sway with Throw and Lift.

The gestures that are similar have not been classified properly in most of the cases. Not to mention that two of the gestures went unrecognized (marked blue) during one of the validation tests (table 8). There can be two reasons behind this – one is that the current classifier parameters, such as number of hidden layers for ANN or selection of SVM kernel, are not suitable for this type of datasets. The other one would be the nature of the data itself. Since the classifiers showed acceptable performance for the generalized cases (above 90% validation set accuracy), we are inclined to believe that we might not be analyzing the data properly. In particular, we believe that it might be an issue of feature selection. Keeping this in mind, our next step was to reduce the dimension of the dataset (it can also be called the feature dataset at this stage) and keep only the data that contribute to most of the variations.

4.2.8 Dimensionality Reduction

As mentioned in section 3.2.5, we calculated 115 features from the raw data. However, all of these features do not contribute well to every gesture make in terms of variation in data. For example, most of the features calculated from sensors 15 and 16 would not be too useful to classify left hand gestures because these two sensors were subject to very limited

to no movement during the performance of those gestures. Likewise, sensors 17 and 18 might confuse the classifier with unnecessary data while it classifies right hand gestures. On the other hand, the participants did not need to move sensor 19, which was placed on their abdomens, to perform gestures apart from Sway. As a result, data from sensor 19 might be very useful to distinguish Sway. Therefore, we need to apply a proper dimensionality reduction technique to get the most out of our feature set.

We decided to use Principal Component Analysis (PCA) here. It is important to mention that PCA is an unsupervised technique [40] whereas our study follows a supervised learning methodology. The way we incorporated PCA into our study was by applying it in batches (similar to performing standardization in batches as described in section 3.2.6) so that the cross-validation or test sets use the same PCA parameters which were used for the training set. The operating principles of PCA have been explained in section 2.5.

To find out the principal components that contribute to most of the variation in the dataset, we sorted the Eigenvectors by decreasing Eigenvalues and selected the PCs with higher Eigenvalues. We implemented this method by using Weka's [54] attribute selection tool which ranks the PCs in terms of their Eigenvalues and keeps only those that fall within the specified variation. Initially, we applied PCA on the entire feature dataset, ranked the PCs using Weka and kept those components which cumulatively attributed to 95% of the variation in the dataset. We followed this procedure for both the generalized and the user specific cases. This is how the variation of data in 2 of the higher ranked principal components look like:

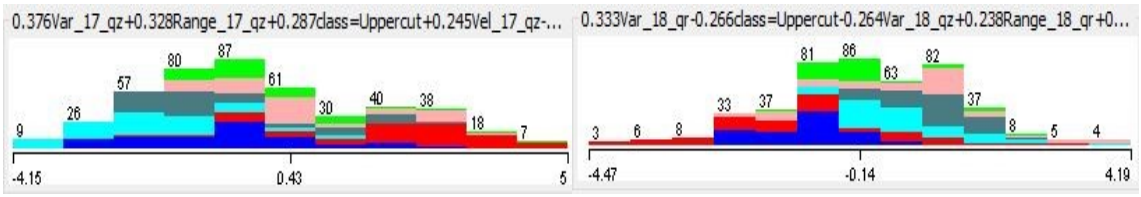


Figure 16: Two of the Principal Components extracted from the feature set

Similar to figure 10, different colour bars indicate different classes or gestures in figure 16. The figure shows that different Principal Components (PC) have different class separations. For instance, if we look at the figure closely, the first PC shows good separation between the blue class and the orange class whereas the second PC shows good separation between blue and teal. Thus, with a combination of different PCs, we are able to provide our classifiers with good separations among all of our classes.

4.2.9 Cross-Validation, Testing and Result Analysis

In this stage, we applied 10-fold cross-validation on our validation sets. As the validation sets and the test sets hold same number of samples, we took a very conservative approach here to make sure that our methodology generalizes well even if the classifiers are given a very small amount of data. We will start with the generalized case and then move forward to the user specific case following the data partition hierarchy as shown in figures 8 and 9.

1. Generalized Gesture Recognition

Table 10 lists the training, cross-validation (for one partition) and test set accuracies of the classifier over different partitions of the dataset (please see figure 9).

Table 10: Training, Validation and Test set accuracies (generalized, after PCA)

Partition	Classifier	Partition	Left Hand Accuracy (%)	Right Hand Accuracy (%)	Combined Dataset Accuracy (%)
60 – 20 - 20	SVM	Training	100	100	N/A
		CV	99.7	100	N/A
		Test	**99.1**	**99.0**	N/A
	ANN	Training	100	100	N/A
		CV	99.7	99.7	N/A
		Test	**99.1**	**99.0**	N/A
70 - 30	SVM	Training	100	100	99.9
		Test	**100**	**100**	**99.7**
	ANN	Training	100	100	99.9
		Test	**100**	**99.3**	**99.7**
60 - 40	SVM	Training	100	100	100
		Test	**99.5**	**100**	**100**
	ANN	Training	100	100	100
		Test	**99.0**	**100**	**100**

* N/A = Not Applicable

From the table above, it is clear that the classifiers achieved near perfect accuracies on the test sets after PCA was applied. Two of the confusion tables (table 11 and 12) for the Left Hand (partition 60-20-20 with SVM) test set is shown below to identify where the classifier became confused. Other confusion tables for these test sets are added in Appendix C.

Table 11: Confusion matrix (in %) for test set accuracy of Left Hand (60-20-20), SVM

	Jab	**Uppercut**	**Throw**	**Lift**	**Block**	**Sway**
Jab	**100**	0	0	0	0	0
Uppercut	0	**95.2**	4.8	0	0	0
Throw	0	0	**100**	0	0	0
Lift	0	0	0	**100**	0	0
Block	0	0	0	0	**100**	0
Sway	0	0	0	0	0	**100**

Table 12: Confusion matrix (in %) for test set accuracy of Right Hand (60-20-20), SVM

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**94.4**	0	5.6	0	0	0
Uppercut	0	**100**	0	0	0	0
Throw	0	0	**100**	0	0	0
Lift	0	0	0	**100**	0	0
Block	0	0	0	0	**100**	0
Sway	0	0	0	0	0	**100**

The average test set accuracies obtained from table 10 are:

SVM:

- Left Hand Gestures = 99.56%
- Right Hand Gestures = 99.67%
- Combined = 99.85%

ANN:

- Left Hand Gestures = 99.85%
- Right Hand Gestures = 99.45%
- Combined = 99.85%

We can clearly see that the classifiers were able to classify almost all of the test set samples, with negligible misclassifications. The average test set accuracies shown above tell us that both SVM and ANN performed very well showing almost identical accuracies. When it comes down to training time, results shown in section 3.2.7 tell us that ANN takes much longer to train than an SVM. However, it does not hold much significance to our study.

2. User Specific Gesture Recognition

Similar to the previous case, we used 10-fold cross-validation to obtain validation set accuracies. The training and validation set accuracies for this experiment is shown below:

Table 13: Training and Validation set accuracies (user specific, after PCA)

Test Case	P1			
Classifier	SVM		ANN	
Accuracy → Dataset ↓	Training	**Validation**	Training	**Validation**
Left Hand	99.76	**9.29**	99.76	**12.65**
Right Hand	100	**12.73**	100	**7.27**
Combined	99.87	**10**	99.87	**15.45**

The current configuration of the experiment yielded in extremely poor validation accuracies for the user specific test. Although we expected the system not to perform as good as it did on the generalized dataset, we did not expect such poor results. Upon further investigation, we found that we overlooked a critical property of this dataset before deciding to use the same configuration for both the cases. Every person has her/his unique way of performing a gesture, even though s/he follows the same instructions. During our study, some of the participants performed the gestures in quick successions whereas others were comparatively slower. As a result, while some of them might have performed 5 to 6 instances of the same gesture within the duration of one sample (4.5 to 5.5 seconds), others might have performed 4-5 gestures within the same period. With this in mind, we made an assumption that keeping velocity based features (velocity and angular velocity) in the dataset confused the classifiers because every participant's speed in performing the gestures varied significantly. This phenomenon basically made everyone's gesture unique. To the classifiers, even two Jabs might look different because they were performed by different individuals which might have resulted in the severe performance drops.

To overcome this problem, we decided to remove the 20 velocity-based features and 15 angular velocity-based features from all of the datasets, standardize the rest of the feature

dataset again following the same procedure as in the generalized case and perform PCA on only on these features. This further reduced the dimensions of the datasets too. Below are the test set results that we obtained after following this procedure. The confusion matrices for the test sets are added in Appendix C.

The table below shows that in most of the cases, we achieved better test set accuracy than training or validation set accuracy. These training and cross-validation sets are actually similar to those under the generalized category as they have data from several participants.

Table 14: Training, Validation and Test set accuracies (user specific, PCA and velocity removal)

Test Case	Classifier	Partition	Left Hand Accuracy (%)	Right Hand Accuracy (%)	Combined Dataset Accuracy (%)
P1	SVM	Training	99.67	100	99.82
		Validation	98.26	100	100
		Test	**100**	**100**	**100**
	ANN	Training	99.67	99.63	99.82
		Validation	99.13	100	100
		Test	**100**	**100**	**100**
P7	SVM	Training	99.73	100	99.82
		Validation	99.75	90	86.79
		Test	**100**	**100**	**100**
	ANN	Training	99.45	100	99.86
		Validation	100	100	91.04
		Test	**100**	**100**	**100**
P11	SVM	Training	99.74	100	99.86
		Validation	89.01	88.61	88.82
		Test	**100**	**88.89**	**100**
	ANN	Training	99.74	100	99.86
		Validation	89.01	96.2	90
		Test	**98.33**	**100**	**99.12**

The table above shows that the changes made by removing velocity and angular velocity-based features had a positive impact on classifier performance and we obtained near perfect to perfect accuracies for all of the cases. This proves that our assumption was true - velocity

and angular velocity should not be included in datasets that are used to classify gestures performed by a specific individual.

5 Chapter: Conclusion

5.1 Overview of Thesis Results

From the results of our experiments, we can deduce that human gesture recognition is not a problem that can be solved using any out of the box classification system. Different scenarios demand different configurations of the experiment and different approach strategies for accurate classification.

We built a complete human gesture recognition system based on Support Vector Machine and Artificial Neural Network for six different gestures by collecting data from 11 participants. We explored two scenarios by organizing the same dataset in two different ways: Generalized Gesture Recognition, where we included data from every individual in our test sets to test our system's performance on recognizing the gestures, regardless of who performed the gestures, and User Specific Gesture Recognition where we tested if our system can perform well when it is given a test dataset from a specific individual to mimic a real life use of the system.

Our experiments revealed that if we have a good set of features, it is easier to recognize random human gestures in general as compared to recognizing a set of gestures from an individual alone. While achieving very good accuracy for the former requires applying basic data preprocessing techniques and common dimensionality reduction methods that are most commonly used in the literature, achieving the same results in the latter scenario is trickier. It requires good understanding of the dataset and proper feature selection methods. We achieved near perfect gesture recognition in the generalized case by following standard experimental methodologies such as feature extraction, standardization, cross-validation and dimensionality reduction using PCA. However, the same methodology

performed poorly on the user specific case. To overcome this problem, we decided to exclude all velocity-based features from our feature set and then follow the same gesture recognition procedure as mentioned above. Eventually, we were able to achieve near perfect overall recognition rates on all of our datasets for this case.

In the beginning of this thesis, we posed two research questions. We believe we were able to answer both of those correctly. The first question was if multiple sensors improve recognition accuracy. Our recognition system has shown improved accuracies in the user specific case as compared to that of our previous study [7] (it was termed as User Excluded). With near perfect recognition rates, we were able to improve the overall accuracy by 5.3% (taking SVM's performance on the combined dataset). This proves that multiple sensors substantially improve recognition accuracy. The other question was which classifier performs better between the two. From table 4, we can see that SVM outperforms ANN in computational efficiency. In terms of accuracy, both the classifiers showed similar performance. Therefore, taking the overall performance into account, we can say it is better to use SVM because it will not only save computation time, but it is also simpler to use. In addition, less computation time may provide the infrastructure for a less expensive, real time recognition system.

In the course of the study, we collected data to build our own quaternion-based human gesture dataset using IMU data, created a complete Euler angle-based dataset from original one and built a feature set comprising 115 features based on 5 base features. The dataset is public and all of the codes are open source. We also developed our own noise removal filter which can effectively remove noise without compromising the overall quality of data. However, it's computational efficiency drops considerably when exposed to a dataset with

very high noise.

5.2 Comparison with the Previous Study

Table 15 below provides a comparison of the findings between our study and a previously done study using a single sensor-based system with the same IMU sensor.

Table 15: Comparison between the previously done single sensor-based system and our system

Previous Study	Our Study
Single sensor	Multiple sensor
3000 instances of gestures	Around 5940 instances of gestures from 1080 samples
Segmented gestures	Unsegmented gestures
Generalized case average* accuracy was 99.70%	Generalized case average* accuracy is 99.85%
User specific case average* accuracy was 94.35%	User specific case average* accuracy was 99.85%

* Using data from gestures common to both studies

Here, the most significant comparison to look at are the differences in recognition accuracies between the two studies. In both the cases, the p-values at a 95% confidence interval was lesser than 0.05 which rejects the null hypothesis that we made a significant improvement in terms of recognition accuracy. However, it is worth mentioning that our system is more complex than a single-sensor based system. Therefore, achieving a higher recognition accuracy in the user specific case means we have a fairly robust model to carry the research forward.

5.3 Limitations and Future Work

Currently, one of the limitations of this system is uninterrupted power supply. The IMUs can be powered by two different sources of energy. They can either be plugged into a computer or other devices to supply power from USB ports or they can run through battery power. The first one restricts mobility and limits portability whereas the second one is not a reliable source of uninterrupted energy. Using rechargeable battery cells with higher energy capacity (mAh rating) may improve this situation but it would make the whole system expensive. The sensors have a few flaws in their design which restricts worry-free handling. The power/reset button on these devices protrude outward which makes it prone to being pressed if the user is not aware of it while performing gestures. We are also trying to design better cases or containers for the sensors so that they can be easily strapped to a participant's body. Lastly, the relatively larger size of the sensors would most likely prevent researchers from using it for gesture recognition using fingers. Another limitation of these sensors is sensor drift which contributes to varying initial accuracy if the issue is not addressed properly. This has been discussed briefly in section 4.1.5. For a detailed explanation, the reader can explore [7].

There can be several possible future applications of our work. The velocity-based features can be used to perform a user validation study that would recognize different users based on the way they performed the gestures. A very important application of our research would be industrial process automation where industrial robots can be trained to mimic human gestures to perform heavy lifting and do other tasks that are otherwise difficult and dangerous for humans. While similar systems are already in use in some of the places, they can mostly perform specific, pre-designed tasks. Other workers in those factories still have

to perform the dangerous but more sophisticated tasks by hand. As a future work, our aim is to carry this research forward and integrate it into a Robotic System.

Appendices

Appendix A : Feature Set

A complete list of features:

Var_15_qr	Range_16_qx	Vel_17_qy	Cov_15_16_qr	Cov_16_19_qx
Var_15_qx	Range_16_qy	Vel_17_qz	Cov_15_16_qx	Cov_16_19_qy
Var_15_qy	Range_16_qz	Vel_18_qr	Cov_15_16_qy	Cov_16_19_qz
Var_15_qz	Range_17_qr	Vel_18_qx	Cov_15_16_qz	Cov_17_18_qr
Var_16_qr	Range_17_qx	Vel_18_qy	Cov_15_17_qr	Cov_17_18_qx
Var_16_qx	Range_17_qy	Vel_18_qz	Cov_15_17_qx	Cov_17_18_qy
Var_16_qy	Range_17_qz	Vel_19_qr	Cov_15_17_qy	Cov_17_18_qz
Var_16_qz	Range_18_qr	Vel_19_qx	Cov_15_17_qz	Cov_17_19_qr
Var_17_qr	Range_18_qx	Vel_19_qy	Cov_15_18_qr	Cov_17_19_qx
Var_17_qx	Range_18_qy	Vel_19_qz	Cov_15_18_qx	Cov_17_19_qy
Var_17_qy	Range_18_qz	Precession_15	Cov_15_18_qy	Cov_17_19_qz
Var_17_qz	Range_19_qr	Nutation_15	Cov_15_18_qz	Cov_18_19_qr
Var_18_qr	Range_19_qx	Spin_15	Cov_15_19_qr	Cov_18_19_qx
Var_18_qx	Range_19_qy	Precession_16	Cov_15_19_qx	Cov_18_19_qy
Var_18_qy	Range_19_qz	Nutation_16	Cov_15_19_qy	Cov_18_19_qz
Var_18_qz	Vel_15_qr	Spin_16	Cov_15_19_qz	
Var_19_qr	Vel_15_qx	Precession_17	Cov_16_17_qr	
Var_19_qx	Vel_15_qy	Nutation_17	Cov_16_17_qx	
Var_19_qy	Vel_15_qz	Spin_17	Cov_16_17_qy	
Var_19_qz	Vel_16_qr	Precession_18	Cov_16_17_qz	
Range_15_qr	Vel_16_qx	Nutation_18	Cov_16_18_qr	
Range_15_qx	Vel_16_qy	Spin_18	Cov_16_18_qx	
Range_15_qy	Vel_16_qz	Precession_19	Cov_16_18_qy	
Range_15_qz	Vel_17_qr	Nutation_19	Cov_16_18_qz	
Range_16_qr	Vel_17_qx	Spin_19	Cov_16_19_qr	

Appendix B : Training and cross-validation set accuracies for different hidden layers

B.1 Generalized Gesture Recognition

Table 16: ANN performance for different hidden layers (generalized)

Left Hand			Right Hand		
Hidden layers	**Training Accuracy (%)**	**CV Accuracy (%)**	**Hidden layers**	**Training Accuracy (%)**	**CV Accuracy (%)**
1	44.1176	40.8696	1	58.8816	47.6636
2	78.8235	69.5652	2	89.4737	71.028
3	97.9412	88.6957	3	94.7368	83.1776
4	95.2941	87.8261	4	92.7632	81.3084
5	97.6471	92.1739	5	97.3684	82.243
6	98.5294	90.4348	6	98.0263	88.785
7	97.3529	92.1739	7	97.3684	86.9159
8	98.2353	90.4348	8	97.0395	85.9813
9	99.4118	93.0435	9	97.6974	90.6542
10	97.9412	92.1739	10	98.0263	90.6542
11	97.6471	90.4348	11	97.6974	91.5888
12	97.9412	91.3043	12	98.0263	85.9813
13	97.3529	93.0435	13	98.0263	88.785
14	97.6471	93.0435	14	97.6974	89.7196
15	**97.6471**	**93.913**	**15**	**98.3553**	**90.6542**
16	97.9412	93.0435	16	97.6974	87.8505
17	98.2353	92.1739	17	97.6974	90.6542
18	98.5294	92.1739	18	98.0263	88.785
19	97.9412	93.0435	19	97.6974	87.8505
20	97.6471	93.0435	20	97.6974	90.6542
60	97.6471	91.3043	60	97.6974	88.785
Average	97.8173	91.7620	Average	97.3338	87.9488

B.2 User Specific Gesture Recognition

Table 17: ANN performance for different hidden layers (user specific)

Left Hand			Right Hand		
Hidden layers	**Training Accuracy (%)**	**CV Accuracy (%)**	**Hidden layers**	**Training Accuracy (%)**	**CV Accuracy (%)**
1	60.6061	33	1	60	38.3929
2	91.2121	34	2	90.411	57.1429
3	98.1818	44	3	98.9041	34.8214
4	98.7879	59	4	99.1781	43.75
5	99.0909	55	5	98.3562	42.8571
6	99.0909	50	6	99.4521	38.3929
7	99.3939	63	7	98.9041	29.4643
8	99.3939	49	8	99.1781	33.9286
9	99.0909	47	9	99.1781	39.2857
10	99.697	49	10	99.4521	42.8571
11	99.697	57	11	99.4521	35.7143
12	100	62	12	99.1781	38.3929
13	100	55	13	99.1781	38.3929
14	99.697	50	14	99.4521	37.5
15	**99.697**	**49**	**15**	**99.4521**	**36.6071**
16	99.697	49	16	99.1781	37.5
17	99.697	47	17	99.4521	33.0357
18	99.697	53	18	99.4521	36.6071
19	99.697	50	19	99.1781	35.7143
20	100	58	20	99.726	36.6071
60	99.697	51	60	99.4521	33.9286
Average	99.4896	52.4736	Average	99.2502	37.1240

Appendix C : Confusion Matrices

C.1 Generalized Gesture Recognition – Initial Experiment

Table 18: Confusion matrix (in %) for validation set accuracy of LH (70-30), ANN

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**95.2**	0	4.8	0	0	0
Uppercut	0	**100**	0	0	0	0
Throw	9.5	0	**90.5**	0	0	0
Lift	0	5.2	0	**84.2**	10.6	0
Block	0	0	0	0	**94.7**	5.3
Sway	0	0	0	0	0	**100**

Table 19: Confusion matrix (in %) for validation set accuracy of case RH (70-30), SVM

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**84.2**	0	0	0	0	15.8
Uppercut	0	**90.0**	10.0	0	0	0
Throw	0	0	**100**	0	0	0
Lift	0	0	0	**100**	0	0
Block	0	0	0	5.6	**94.4**	0
Sway	0	0	0	0	0	**100**

C.2 User Specific Gesture Recognition – Initial Experiment

Table 20: Confusion matrix (in %) for validation set accuracy of case LH, ANN

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**0.0**	0	47.0	53.0	0	0
Uppercut	0	**0.0**	0	0	100	0
Throw	50.0	0	**50.0**	0	0	0
Lift	0	0.0	0	**65.0**	35.0	0
Block	0	0	0	60.0	**40.0**	5.3
Sway	0	0	33.3	0	0	**66.7**

Table 21: Confusion matrix (in %) for validation set accuracy of case RH, SVM

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**73.3**	0	26.7	0	0	0
Uppercut	50.0	**38.9**	0	0	11.1	0
Throw	0	0	**100**	0	0	0
Lift	0	0	5.55	**88.9**	5.55	0
Block	0	0	66.7	5.5	**27.8**	0
Sway	0	0	93.0	0	0	**7.0**

C.3 Generalized Gesture Recognition – PCA

Table 22: Confusion matrix (in %) for test set accuracy of Left Hand (60-20-20), ANN

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**100**	0	0	0	0	0
Uppercut	0	**95.2**	4.8	0	0	0
Throw	0	0	**100**	0	0	0
Lift	0	0	0	**100**	0	0
Block	0	0	0	0	**100**	0
Sway	0	0	0	0	0	**100**

Table 23*: Confusion matrix (in %) for test set accuracy of Left Hand (70-30), SVM

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**100**	0	0	0	0	0
Uppercut	0	**100**	0	0	0	0
Throw	0	0	**100**	0	0	0
Lift	0	0	0	**100**	0	0
Block	0	0	0	0	**100**	0
Sway	0	0	0	0	0	**100**

Table 24*: Confusion matrix (in %) for test set accuracy of Left Hand (70-30), ANN

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**100**	0	0	0	0	0
Uppercut	0	**100**	0	0	0	0
Throw	0	0	**100**	0	0	0
Lift	0	0	0	**100**	0	0
Block	0	0	0	0	**100**	0
Sway	0	0	0	0	0	**100**

Table 25: Confusion matrix (in %) for test set accuracy of Left Hand (60-40), SVM

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**100**	0	0	0	0	0
Uppercut	0	**97.6**	0	0	0	2.4
Throw	0	0	**100**	0	0	0
Lift	0	0	0	**100**	0	0
Block	0	0	0	0	**100**	0
Sway	0	0	0	0	0	**100**

Table 26: Confusion matrix (in %) for test set accuracy of case Left Hand (60-40), ANN

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**100**	0	0	0	0	0
Uppercut	0	**97.6**	0	0	0	2.4
Throw	0	0	**100**	0	0	0
Lift	0	0	0	**100**	0	0
Block	0	0	0	0	**100**	0
Sway	0	0	0	0	0	**100**

Table 27: Confusion matrix (in %) for test set accuracy of Right Hand (60-20-20), ANN

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**94.4**	0	5.6	0	0	0
Uppercut	0	**100**	0	0	0	0
Throw	0	0	**100**	0	0	0
Lift	0	0	0	**100**	0	0
Block	0	0	0	0	**100**	0
Sway	0	0	0	0	0	**100**

Table 28*: Confusion matrix (in %) for test set accuracy of Right Hand (70-30), SVM

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**100**	0	0	0	0	0
Uppercut	0	**100**	0	0	0	0
Throw	0	0	**100**	0	0	0
Lift	0	0	0	**100**	0	0
Block	0	0	0	0	**100**	0
Sway	0	0	0	0	0	**100**

Table 29: Confusion matrix (in %) for test set accuracy of Right Hand (70-30), ANN

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**100**	0	0	0	0	0
Uppercut	0	**96.6**	0	0	0	3.4
Throw	0	0	**100**	0	0	0
Lift	0	0	0	**100**	0	0
Block	0	0	0	0	**100**	0
Sway	0	0	0	0	0	**100**

Table 30*: Confusion matrix (in %) for test set accuracy of Right Hand (60-40), SVM

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**100**	0	0	0	0	0
Uppercut	0	**100**	0	0	0	0
Throw	0	0	**100**	0	0	0
Lift	0	0	0	**100**	0	0
Block	0	0	0	0	**100**	0
Sway	0	0	0	0	0	**100**

Table 31: Confusion matrix (in %) for test set accuracy of Right Hand (60-40), ANN

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**100**	0	0	0	0	0
Uppercut	0	**100**	0	0	0	0
Throw	0	0	**100**	0	0	0
Lift	0	0	0	**100**	0	0
Block	0	0	0	0	**100**	0
Sway	0	0	0	0	0	**100**

Table 32: Confusion matrix (in %) for test set accuracy of Combined (70-30), SVM

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**100**	0	0	0	0	0
Uppercut	0	**100**	0	0	0	0
Throw	0	0	**100**	0	0	0
Lift	0	0	0	**100**	0	0
Block	0	0	0	0	**98.2**	1.8
Sway	0	0	0	0	0	**100**

Table 33: Confusion matrix (in %) for test set accuracy of Combined (70-30), ANN

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**100**	0	0	0	0	0
Uppercut	0	**100**	0	0	0	0
Throw	0	0	**100**	0	0	0
Lift	0	0	0	**100**	0	0
Block	0	0	0	0	**98.2**	1.8
Sway	0	0	0	0	0	**100**

Table 34: Confusion matrix (in %) for test set accuracy of Combined (60-40), SVM

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**100**	0	0	0	0	0
Uppercut	0	**100**	0	0	0	0
Throw	0	0	**100**	0	0	0
Lift	0	0	0	**100**	0	0
Block	0	0	0	0	**100**	0
Sway	0	0	0	0	0	**100**

Table 35: Confusion matrix (in %) for test set accuracy of Combined (60-40), ANN

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**100**	0	0	0	0	0
Uppercut	0	**100**	0	0	0	0
Throw	0	0	**100**	0	0	0
Lift	0	0	0	**100**	0	0
Block	0	0	0	0	**100**	0
Sway	0	0	0	0	0	**100**

C.4 User Specific Gesture Recognition – PCA, Velocity-based Features Removed

Table 36: Confusion matrix (in %) for test set accuracy of Left Hand, ANN

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**100**	0	0	0	0	0
Uppercut	0	**100**	0	0	0	0
Throw	0	0	**100**	0	0	0
Lift	0	0	0	**100**	0	0
Block	0	0	0	0	**90**	10
Sway	0	0	0	0	0	**100**

Table 37: Confusion matrix (in %) for test set accuracy of Combined, ANN

	Jab	Uppercut	Throw	Lift	Block	Sway
Jab	**100**	0	0	0	0	0
Uppercut	0	**100**	0	0	0	0
Throw	0	0	**100**	0	0	0
Lift	0	0	0	**100**	0	0
Block	0	0	0	0	**94.7**	5.3
Sway	0	0	0	0	0	**100**

Appendix D : Research Ethics Clearance

This research has been approved by Carleton University Research Ethics Board (CUREB):

Study number: 15-129
Principal Investigator: Shamir Alavi
Co-Investigator (Faculty Supervisor): Anthony Whitehead
Department: Systems and Computer Engineering
Approval date: 2015/08/24
Expiry date: 2016/08/31

References

[1] S. Mitra and T. Acharya, "Gesture recognition : A Survey," *IEEE Trans. Syst. Man, Cybern. - Part C Appl. Rev.*, vol. 37, no. 3, pp. 311–324, 2007.

[2] F. G. Hofmann, P. Heyer, and G. Hommel, "Velocity profile based recognition of dynamic gestures with discrete Hidden Markov Models," *Gesture Sign Lang. Human-Computer Interact.*, vol. 1371, pp. 81–95, 1998.

[3] T. B. Moeslund and E. Granum, "A survey of Computer Vision-based human motion capture," *Comput. Vis. Image Underst.*, vol. 81, no. 3, pp. 231–268, 2001.

[4] J.-H. Kim, N. D. Thang, and T.-S. Kim, "3-D hand motion tracking and gesture recognition using a data glove," *Ind. Electron. 2009. ISIE 2009. IEEE Int. Symp.*, no. ISlE, pp. 1013–1018, 2009.

[5] X. Zhang, X. Chen, Y. Li, V. Lantz, K. Wang, and J. Yang, "A framework for hand gesture recognition based on accelerometer and EMG sensors," *IEEE Trans. Syst. Man, Cybern. Part ASystems Humans*, vol. 41, no. 6, pp. 1064–1076, 2011.

[6] S. W. S. Wang, J. Y. J. Yang, N. C. N. Chen, X. C. X. Chen, and Q. Z. Q. Zhang, "Human activity recognition with user-free accelerometers in the sensor networks," *2005 Int. Conf. Neural Networks Brain*, vol. 2, pp. 1212–1217, 2005.

[7] D. Arsenault, "A quaternion-based motion tracking and gesture recognition system using wireless inertial sensors." Carleton University, Ottawa, 2014.

[8] A. Mannini and A. M. Sabatini, "Machine learning methods for classifying human physical activity from on-body accelerometers," *Sensors*, vol. 10, no. 2, pp. 1154–1175, 2010.

[9] C. Zhu and W. Sheng, "Wearable sensor-based hand gesture and daily activity recognition for robot-assisted living," *IEEE Trans. Syst. Man, Cybern. Part ASystems Humans*, vol. 41, no. 3, pp. 569–573, 2011.

[10] J. Mantyjarvi, J. Himberg, and T. Seppanen, "Recognizing human motion with multiple acceleration sensors," *2001 IEEE Int. Conf. Syst. Man Cybern. e-Systems e-Man Cybern. Cybersp.*, vol. 2, pp. 2–7, 2001.

[11] T. Schlömer, B. Poppinga, N. Henze, and S. Boll, “Gesture recognition with a Wii controller,” *Proc. 2nd Int. Conf. Tangible Embed. Interact. TEI 08*, p. 11, 2008.

[12] S. Reifinger, F. Wallhoff, M. Ablassmeier, T. Poitschke, and G. Rigoll, “Static and dynamic hand-gesture recognition for augmented reality applications,” *Human-Computer Interact. Pt 3, Proceedings;4552 728-737 2007*, vol. 4552, pp. 728–737, 2007.

[13] Y. Wu and T. S. Huang, “Vision-based gesture recognition: a review,” *Gesture-Based Commun. Human-Computer Interact. Int. GestureWorkshop, GW’99 Gif-sur-Yvette, Fr. March 17-19, 1999 Proc.*, pp. 103–115, 1999.

[14] C. Zhu and W. Sheng, “Human daily activity recognition in robot-assisted living using multi-sensor fusion,” *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 2154–2159, 2009.

[15] P. Neto, J. N. Pires, and a. P. Moreira, “High-level programming and control for industrial robotics: using a hand-held accelerometer-based input device for gesture and posture recognition,” *Ind. Robot An Int. J.*, vol. 37, no. 2, pp. 137–147, 2010.

[16] S. Ullah, H. Higgins, B. Braem, B. Latre, C. Blondia, I. Moerman, S. Saleem, Z. Rahman, and K. S. Kwak, “A comprehensive survey of wireless body area networks,” *J. Med. Syst.*, vol. 36, no. 3, pp. 1065–1094, 2012.

[17] E. S. Choi, W. C. Bang, S. J. Cho, J. Yang, D. Y. Kim, and S. R. Kim, “Beatbox music phone: Gesture-based interactive mobile phone using a tri-axis accelerometer,” *Proc. IEEE Int. Conf. Ind. Technol.*, vol. 2005, pp. 97–102, 2005.

[18] K. Rennie, T. Rowsell, S. a Jebb, D. Holburn, and N. J. Wareham, “A combined heart rate and movement sensor: proof of concept and preliminary testing study.,” *Eur. J. Clin. Nutr.*, vol. 54, no. 5, pp. 409–414, 2000.

[19] T. Westeyn, P. Presti, and T. Starner, “ActionGSR: A combination galvanic skin response-accelerometer for physiological measurements in active environments,” *Proc. - Int. Symp. Wearable Comput. ISWC*, no. September 2015, pp. 129–130, 2007.

[20] X. Tang, Y. Liu, C. Lv, and D. Sun, “Hand motion classification using a multi-channel surface electromyography sensor,” *Sensors*, vol. 12, no. 2, pp. 1130–1147, 2012.

[21] A. Bashashati, M. Fatourechi, R. K. Ward, and G. E. Birch, “A survey of signal processing algorithms in brain-computer interfaces based on electrical brain signals.,” *J. Neural Eng.*, vol. 4, no. 2, pp. R32–R57, 2007.

[22] W. J. Yoo, K. W. Jang, J. K. Seo, J. Y. Heo, J. S. Moon, J. Y. Park, and B. Lee, “Development of respiration sensors using plastic optical fiber for respiratory monitoring inside MRI system,” *J. Opt. Soc. Korea*, vol. 14, no. 3, pp. 235–239, 2010.

[23] A. Whitehead, N. Crampton, K. Fox, and H. Johnston, “Sensor networks as video game input devices,” *Proc. 2007 Conf. Futur. Play - Futur. Play ’07*, p. 38, 2007.

[24] G. Welch and E. Foxlin, “Motion tracking: No silver bullet, but a respectable arsenal,” *IEEE Comput. Graph. Appl.*, vol. 22, no. 6, pp. 24–38, 2002.

[25] J. Hol, Sensor fusion and calibration of inertial sensors, vision, ultra-wideband and GPS, no. 1368, *Linköping University*, Linköping, Sweden, 2011.

[26] A. Kim and M. Golnaraghi, “Initial calibration of an inertial measurement unit using an optical position tracking system,” IEEE *Position Location and Navigation Symposium,* pp. 96–101, 2004.

[27] R. Mukundan, “Quaternions : From classical mechanics to computer graphics , and beyond,” *Proc. 7th Asian Technol. Conf. Math.*, pp. 97–106, 2002.

[28] L. Vicci, “Quaternions and rotations in 3-space: The algebra and its geometric interpretation,” TR01-014, Dept. of Computer Science, UNC Chapel Hill, pp. 1–11, 2001.

[29] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, Second Edi. Wiley, 2000.

[30] M. Chen, S. Mao, Y. Zhang, and V. C. M. Leung, “Big data: related technologies, challenges and future prospects,” *Springer,* pp. 51–58, 2014.

[31] H. Frezza, “Support Vector Machines Tutorial,”, *Available at: http://www.metz.supelec.fr*, 2013.

[32] A. Zisserman, “Lecture 2 : The SVM classifier,”, *Available at: http://www.robots.ox.ac.uk/~az,* 2013.

[33] A. Ng, “Support Vector Machines,” *Available at: cs229.stanford.edu.*, 2012.

[34] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “A training algorithm for optimal margin classifiers,” *Proc. 5th Annu. ACM Work. Comput. Learn. Theory*, pp. 144–152, 1992.

[35] G. C. Cawley and N. L. C. Talbot, “On over-fitting in model selection and subsequent selection bias in performance evaluation,” *J. Mach. Learn. Res.*, vol. 11, p. 2079−2107, 2010.

[36] A. Abraham, “129: Artificial neural networks,” Handbook of measuring system design, *John Wiley & Sons, Ltd*, vol. 346, p. 901-908, 2015.

[37] N. Benvenuto and F. Piazza, “On the complex backpropagation algorithm,” *IEEE Trans. Signal Process.*, vol. 40, no. 4, pp. 967–969, 1992.

[38] S. G. Anantwar and R. R. Shelke, “Simplified Approach of ANN : Strengths and Weakness,” *Int. J. Eng. Innov. Technol.*, vol. 1, no. 4, pp. 73–77, 2012.

[39] I. M. Guyon, S. Gunn, M. Nikravesh, L. Zadeh, "Feature extraction: foundations and applications," *Springer*, Berlin, 2006.

[40] S. Wold, K. Esbensen, and P. Geladi, “Principal component analysis,” *Chemom. Intell. Lab. Syst.*, vol. 2, pp. 37–52, 1987.

[41] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan, “uWave: Accelerometer-

based personalized gesture recognition and its applications," *Pervasive Mob. Comput.*, vol. 5, no. 6, pp. 657–675, 2009.

[42] H. Junker, O. Amft, P. Lukowicz, and G. Tröster, "Gesture spotting with body-worn inertial sensors to detect user activities," *Pattern Recognit.*, vol. 41, no. 6, pp. 2010–2024, 2008.

[43] J.-C. Lementec and P. Bajcsy, "Recognition of arm gestures using multiple orientation sensors: gesture classification," *Proceedings. 7th Int. IEEE Conf. Intell. Transp. Syst. (IEEE Cat. No.04TH8749)*, pp. 965–970, 2004.

[44] P. Maes, T. Darrell, B. Blumberg, and A. Pentland, "The ALIVE system: wireless, full-body interaction with autonomous agents," *Multimed. Syst.*, vol. 5, no. 2, pp. 105–112, 1997.

[45] B. Peng, G. Qian, and S. Rajko, "View-invariant full-body gesture recognition via multilinear analysis of voxel data," *Distrib. Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE Int. Conf.*, pp. 1–8, 2009.

[46] B. Peng, G. Qian, and S. Rajko, "View-invariant full-body gesture recognition from video," *19th Int. Conf. Pattern Recognit.*, pp. 1–5, 2008.

[47] Microchip, "Motion sensing demo board." *Available at:* http://www.microchip.com, 2014.

[48] InvenSense, "MPU-6000/6050 six-axis MEMS motion tracking devices." *Available at: http://www.invensense.com/products/motion-tracking/6-axis*, 2014.

[49] Z.-Q. Zhang, L.-Y. Ji, Z.-P. Huang, and J.-K. Wu, "Adaptive Information Fusion for Human Upper Limb Movement Estimation," *IEEE Trans. Syst., Man, Cybern. {A}*, vol. 42, no. 5, pp. 1100–1108, 2012.

[50] H. J. Luinge and P. H. Veltink, "Measuring orientation of human body segments using miniature gyroscopes and accelerometers," *Med. Biol. Eng. Comput.*, vol. 43, no. 2, pp. 273–282, 2005.

[51] X. Yun and E. R. Bachmann, "Design, Implementation, and Experimental Results of a Quaternion-Based Kalman Filter for Human Body Motion Tracking," *Robot. IEEE Trans.*, vol. 22, no. 6, pp. 1216–1227, 2006.

[52] R. C. Hibbeler, "Engineering mechanics," *Pearson Prentice Hall,* Upper Saddle River, New Jersey, 2009.

[53] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix*, vol. 58, pp. 1–35, 2006.

[54] I. H. W. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, "The WEKA data mining software: an update," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10-18, 2009.